SIEMENS

COB1 (BS2000) COBOL-Compiler

Benutzerhandbuch

Nachtrag August 1986 (Softwareprodukt COB1 V2.3A)

Bestell-Nr. U254-J2-Z55-3
Printed in the Federal Republic of Germany
1730 AG 7864. (2170)

Bestellnummer:
U254-J2-Z55-3

68521

COB1 (BS2000)
COBOL-Compiler

Benutzerhandbuch

Ausgabe November 1984 (Softwareprodukt COB1 V2.2A)
Nachtrag September 1985 (Softwareprodukt COB1 V2.2B)
Nachtrag August 1986 (Softwareprodukt COB1 V2.3A)

Bestell-Nr. U254-J-Z55-1
Printed in the Federal Republic of Germany
2220 AG 11846. (2770)

Bestell-Nr. U254-J1-Z55-2
Printed in the Federal Republic of Germany
1120 AG 9855. (1400)

Bestell-Nr. U254-J2-Z55-3
Printed in the Federal Republic of Germany
1730 AG 7864. (2170)

Siemens Aktiengesellschaft

Vorwort

Dieses Handbuch leitet den Leser an, wie er COBOL-Programme

- im BS2000 erstellt,
- mit dem COB1-Compiler übersetzt,
- mit weiteren Betriebssystemkomponenten verwaltet und bindet
- und Test- und Produktionsläufe durchführt.

Die Kapitel 1 bis 5 beschreiben diesen Weg.

Das Kapitel 6 gibt Programmierhinweise zur

- Testunterstützung durch COBOL-Sprachelemente,
- Erstellung von wiederverwendbaren Moduln (reentrant Code),
- Unterprogrammtechnik,
- Arbeit mit Systemdateien durch COBOL-Anweisungen,
- Dateibearbeitung,
- Anwendung von Sortier- und Mischfunktionen und
- Unterstützung bei der Ausgabe von Fixpunkten sowie der Wiederanlaufbehandlung.

Der Anhang enthält

- die Meldungen des COB1-Systems,
- die Beschreibung des Aufbaus des COB1-Systems und der erzeugten Objektmodule und
- die Beschreibung der Anwendung der COBOL-DML-Sprachelemente.

Der Leser benötigt Kenntnisse der Programmiersprache COBOL. Der Sprachumfang ist im Manual „COB1 Beschreibung" [1] dargestellt.

Auf Druckschriften zu Betriebskomponenten, die für die Entwicklung und den Ablauf eines COBOL-Programms zur Anwendung kommen, wird im Text durch Kurztitel bzw. Nummern in eckigen Klammern hingewiesen. Die vollständigen Titel sind im Literaturverzeichnis aufgeführt.

Programm- und Ablaufbeispiele sind grau unterlegt, darin enthaltene Dialogeingaben rot hervorgehoben.

Die Neuerungen gegenüber dem Vorgängermanual sind im Änderungsprotokoll 1 zusammengefaßt.

Mit Wünschen und Vorschlägen und Ihrer Kritik können Sie zur Verbesserung unserer Manuale beitragen. Bitte benutzen Sie dazu die rosa Formblätter am Ende des Manuals.

Manualredaktion D ST PM 2
Otto-Hahn-Ring 6, 8000 München 83

Änderungsprotokoll 1

Änderung des Manuals, Stand Januar 1984 (Softwareprodukt COB1 V2.1) durch die Neuausgabe vom November 1984 (Version 2.2A)

Die Quellcodeeingabe aus LMS-Programmbibliotheken und die Bindemodulsicherung in LMS-Programmbibliotheken wird unterstützt.

Das Lesen von Quellcode aus LMS-Programmbibliotheken kann hierarchisch aus 5 Bibliotheken erfolgen. Dafür stehen 4 zusätzliche LINK-Namen (COBLIB1 bis COBLIB4) zur Verfügung.

Der neue Zusatz WITH NO LOCK zur READ- und START-Anweisung für indizierte und relative Dateien ermöglicht es, nach OPEN I-O und FILE-Parameter SHARUPD=yes einen Satz zu lesen, ohne den Block, in dem er steht, für andere Anwender zu sperren.

COB1 V2.2A bedient neue COBRUN-Operanden.

- LINK — Bei Angabe dieses Operanden wird der Linkname für im Programm verwendete Dateien aus der ASSIGN-Klausel und nicht aus dem Dateinamen der SELECT-Klausel entnommen.
- MODULE — Der erzeugte Bindemodul wird in die angegebene LMS-Programmbibliothek eingetragen.
- NODDLIST — In der Quellcodeliste wird das SUBSCHEMA für den Datenbank-Anschluß unterdrückt.

Die COBRUN-Operanden ANSICOB, COPY und COBNAM werden nicht mehr unterstützt.

Die folgende Tabelle gibt die Seiten an, die Änderungen enthalten.

Seite	Stichwort	neu	geändert	entfallen
1-19	Beschreibung der LMS-Bibliothekstypen	×		
1-19	Zuweisen von LMS-Programmbibliotheken		×	
2-11	COPY-Anweisung mit Zusatz SUPPRESS	×		
2-14	COBRUN MODULE=bibliotheksname	×		
2-17	Ausgabe von Bindemoduln		×	
2-18	COBRUN MODULE=bibliotheksname	×		
2-19	Fußnote zu COBRUN MODULE	×		
2-19	COBRUN MODULE	×		
2-19	COBRUN COBNAM			×
2-25	Steueranweisungsliste	×		
2-27	Bibliotheksliste	×		
2-30	Bibliotheksliste	×		

Seite	Stichwort	neu	geän-dert	ent-fallen
2-42	Internal Symbol Dictionary		×	
2-44	Internal Symbol Dictionary		×	
2-47	Internal Symbol Dictionary		×	
2-49	Internal Symbol Dictionary		×	
2-55	LINK-name ERRLINK		×	
2-57	COBRUN ANSICOB, COBNAM, COPY			×
2-59	COBRUN MODULE-Hinweis	×		
2-62	COBRUN ANSICOB, COBNAM, COPY			×
2-63	COBRUN LINK, MODULE, NODDLIST	×		
4-3/4	TIME-Operand			×
5-7	COBRUN LINK		×	
5-14	Tabelle bereinigt		×	
5-23 ff	Zugriff auf Übersetzer und Betriebssysteminformationen	×		
6-26	APPLY WRITE ONLY Klausel			×
6-30	Beispiel zu SPECIAL-NAMES	×		
6-35	COBRUN COBNAM			×
6-37	Tabelle um Angabe WITH NO LOCK erweitert	×		
6-38	Aufbau einer relativen Datei		×	
6-40	COBRUN COBNAM			×
6-41	FILE STATUS 94		×	
6-44	Tabelle um Angabe WITH NO LOCK erweitert	×		
6-49 ff	START oder READ mit Zusatz WITH NO LOCK	×		
6-54	Beispiel zu USE AFTER STANDARD ERROR	×		
Anhang	Der Abschnitt Meldungen wurde überarbeitet			

Änderungsprotokoll 2

Änderung des Manuals,
Stand November 1984 (Softwareprodukt COB1 V2.2A)
durch den Nachtrag vom
September 1985 (COB1 V2.2B)

Seite	Stichwort	neu	geän-dert	ent-fällt
1-12 bis 1-18	COBLUR (COBOL-Bibliotheken)			×
2-5	Eingabe aus Bibliotheken		×	
2-8	COBLUR			×
2-9	Beispiel 14		×	
2-10	COBRUN SRCELEM	×		
2-13	COBRUN SEMCHK, SRCELEM	×		
2-14	COBRUN LOW#UP, SSEQ#GEN	×		
2-16	Beispiel 17		×	
2-21	Beispiel 20, Text		×	
2-22	Tabelle		×	
2-25	Listenbeispiel		×	
2-57	Parameter-Übersicht		×	
2-59	PARAM CARD/DISC			×
2-62	END-Anweisung		×	
2-63	COBRUN LOW#UP	×		
2-64	COBRUN SEMCHK, SRCELEM, SSEQ#GEN	×		
5-10	EXEC-Operanden		×	
5-24	Geräte-Typ-Tabelle		×	
6-15	Beenden von Programm-Unterbrechungen			×
6-15	Aufbau des Sicherstellungsbereiches		×	
6-16	Beenden von Programm-Unterbrechungen	×		
6-15	RETURN-CODE	×		
6-26	OPEN-Modus-Wirkung	×		
6-26ff.	Neuordnung der Seiten		×	

Seite	Stichwort	neu	geän- dert	ent- fällt
6-27f.	Sequentielle Dateiorganisation		×	
6-31	FILE STATUS-Wert 39		×	
6-35	COBRUN COBNAM			×
6-41	FILE STATUS-Wert 39		×	
6-48	FILE STATUS-Wert 39		×	
6-58	SORTOUTnn			×
A-1	Allgemeine Angaben zu Fehlermeldungen	×		
A-1c	COBRUN SSEQ#GEN	×		
A-4	FM9023	×		
A-5	FM9026	×		
A-6	FM9040, FM9044	×		
A-11	FM9072A	×		
A-14	FM9096	×		
A-20	ITCOCALO			×

Änderungsprotokoll 3

Änderung des Vorgänger-Manuals,
Stand September 1985 (COB1, V.2.2B),
durch den Nachtrag vom
August 1986 (COB1 V2.3A)

COB1 V2.3A enthält gegenüber der Vorgänger-Version die folgenden wesentlichen Änderungen:

- Anschluß an die Dialogtesthilfe AID
- Neue Fehlerklasse „Hinweise"
- INITIALIZE-Anweisung
- Funktionserweiterungen für Jobvariablen
- Bedienung mehrerer SYSLST-Dateien

Die folgende Tabelle gibt nur Seiten an, die fachliche Änderungen enthalten.

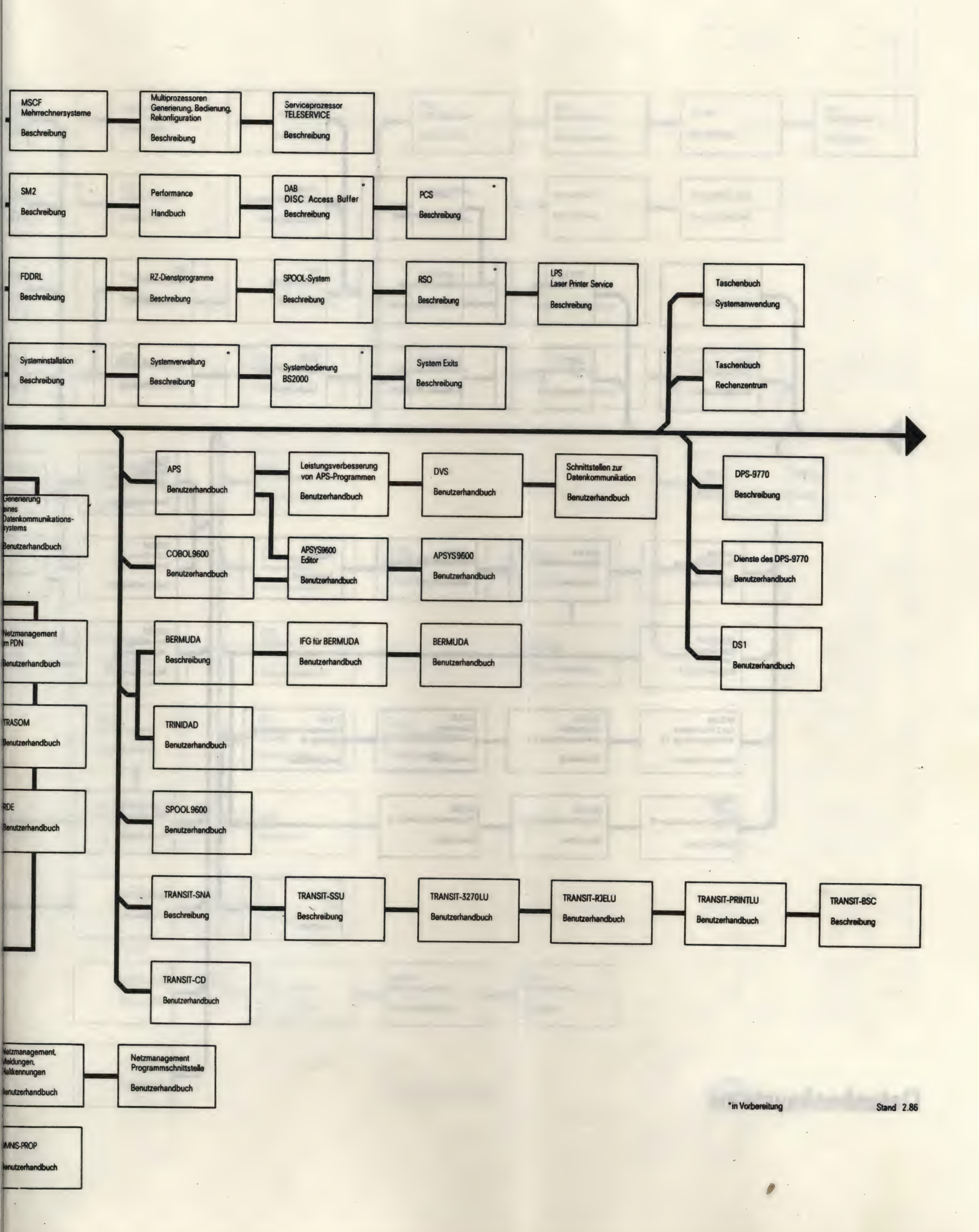
Seite	Stichwort	neu	geän-dert	ent-fallen
1-3	Beispiel 1: COBOL-Quellprogramm		x	
1-5	Beispiel 2: Eingabe eines COBOL-Programmes im Kartenformat			x
1-7—1-10	Eingabe und Änderung von COBOL-Programmen mit EDOR			x
2-14f	COBRUN SYMTEST	x		
2-16	COBRUN ERRPRn		x	
2-25—2-56	Übersetzerlisten		x	
2-31f	Fußnoten zum OBJECT-Listing			x
2-56	Fehlerklasse I (Hinweismeldungen)	x		
2-57	COBRUN SYMTEST	x		
2-58	Beendigungsverhalten des COB1 (Tabelle mit den Return-Codes für Jobvariablen)	x		
2-60	Hinweis zu PARAM SYMDIC=YES	x		
2-62	COBRUN ERRPRn		x	
2-64f	COBRUN SYMTEST	x		
4-3	Operanden des EXEC Kommandos		x	
4-4	Operanden des LOAD-Kommandos		x	
4-6	SYMTEST-Operand in der PROGRAM-Anweisung	x		

Änderungsprotokoll

Seite	Stichwort	neu	geän-dert	ent-fallen
5-1—5-3	SYSLSTnn-Unterstützung	x		
5-8	Beendigungsverhalten eines COBOL-Programmes (Tabelle mit den internen Return-Codes und den zugeordneten „90/91-er" Meldungen)	x		
5-9	Operanden des EXEC-Kommandos		x	
5-11—5-15b	Dialogtesthilfe AID	x		
5-11—5-15	Dialogtesthilfe IDA			x
6-4	Beschreibung der ON-Anweisung			x
6-6	Beschreibung der *DEBUG-Anweisung			x
6-26a—6-26b	Übertragungsarten für Daten in/aus Dateien	x		
6-28	Hinweis zum FILE-Kommando (für RECFORM=U)	x		
6-32—6-36	Beschreibung der direkten Dateiorganisation			x
6-48	FILE STATUS-Wert 94 (Beschreibung)		x	
A-2	FM 9001		x	
A-5	FM 9030	x		
A-6	FM 9031, FM 9032, FM 9033	x		
A-7	FM 9040 (Beschreibung), FM 9047		x	
A-15	FM 9105	x		
A-19—A-21	Moduln des Laufzeitsystems		x	

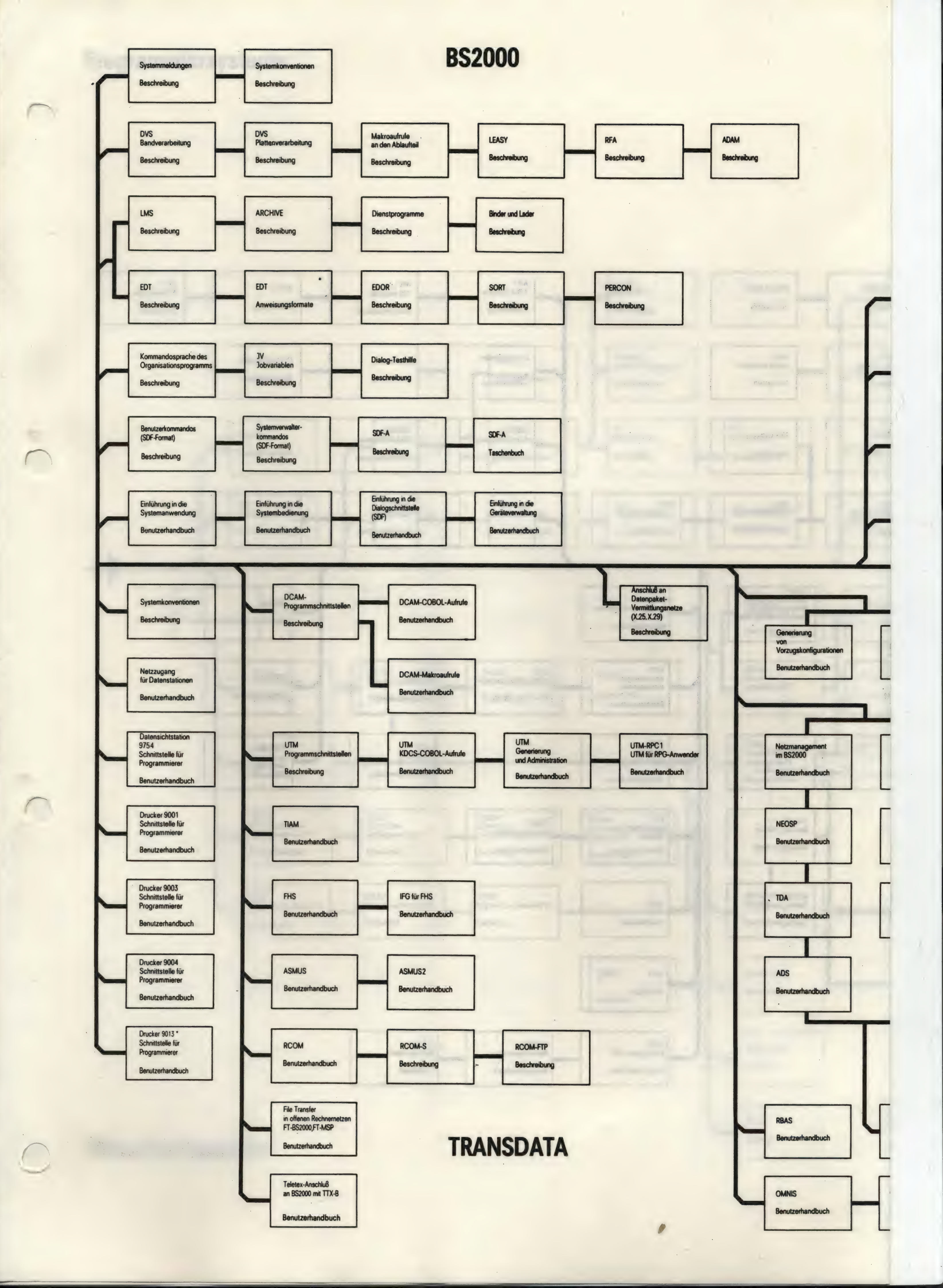
Struktur der Manuale für

- BS2000
- TRANSDATA
- Programmiersysteme
- Datenbanken



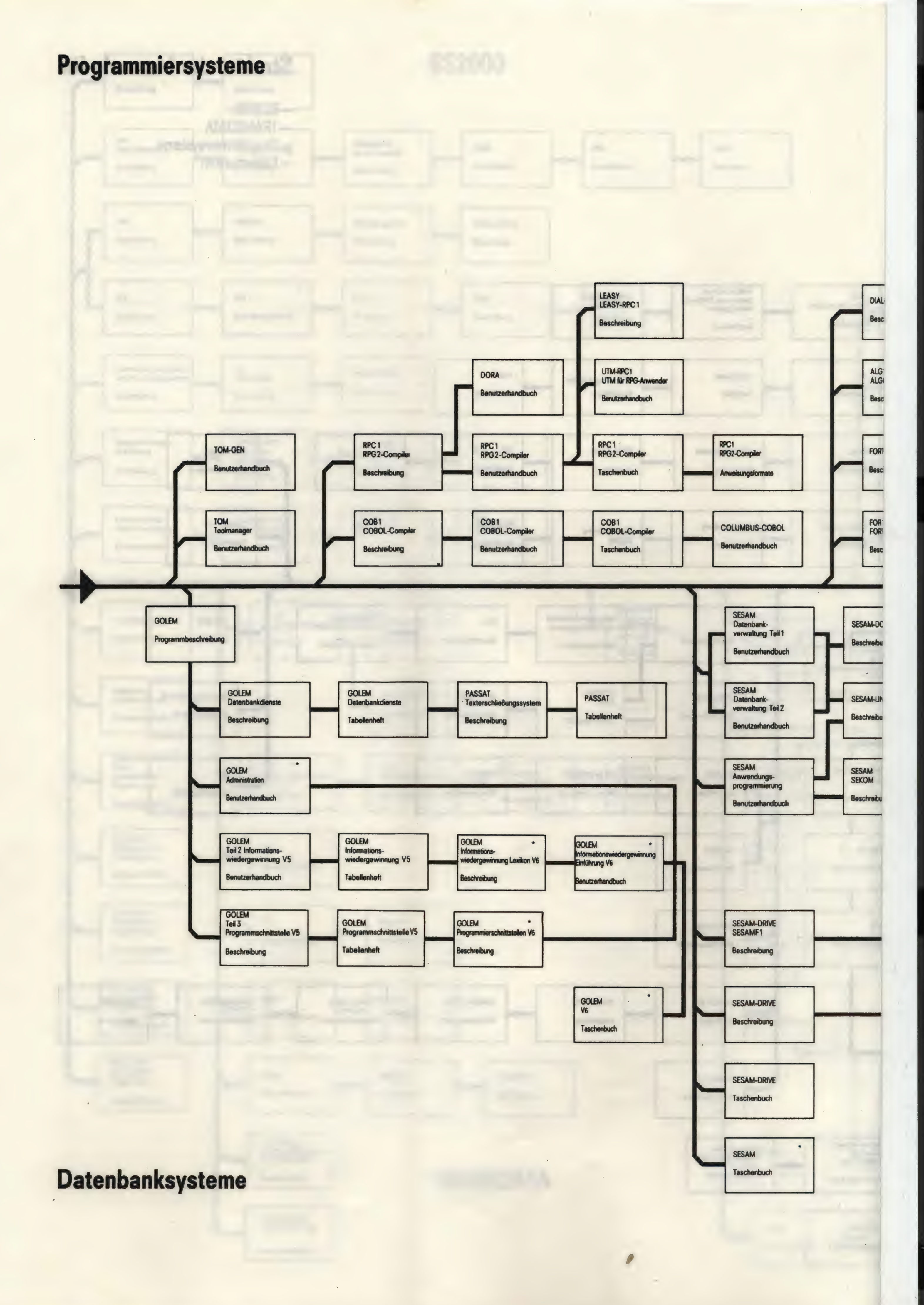
*in Vorbereitung

Stand 2.86

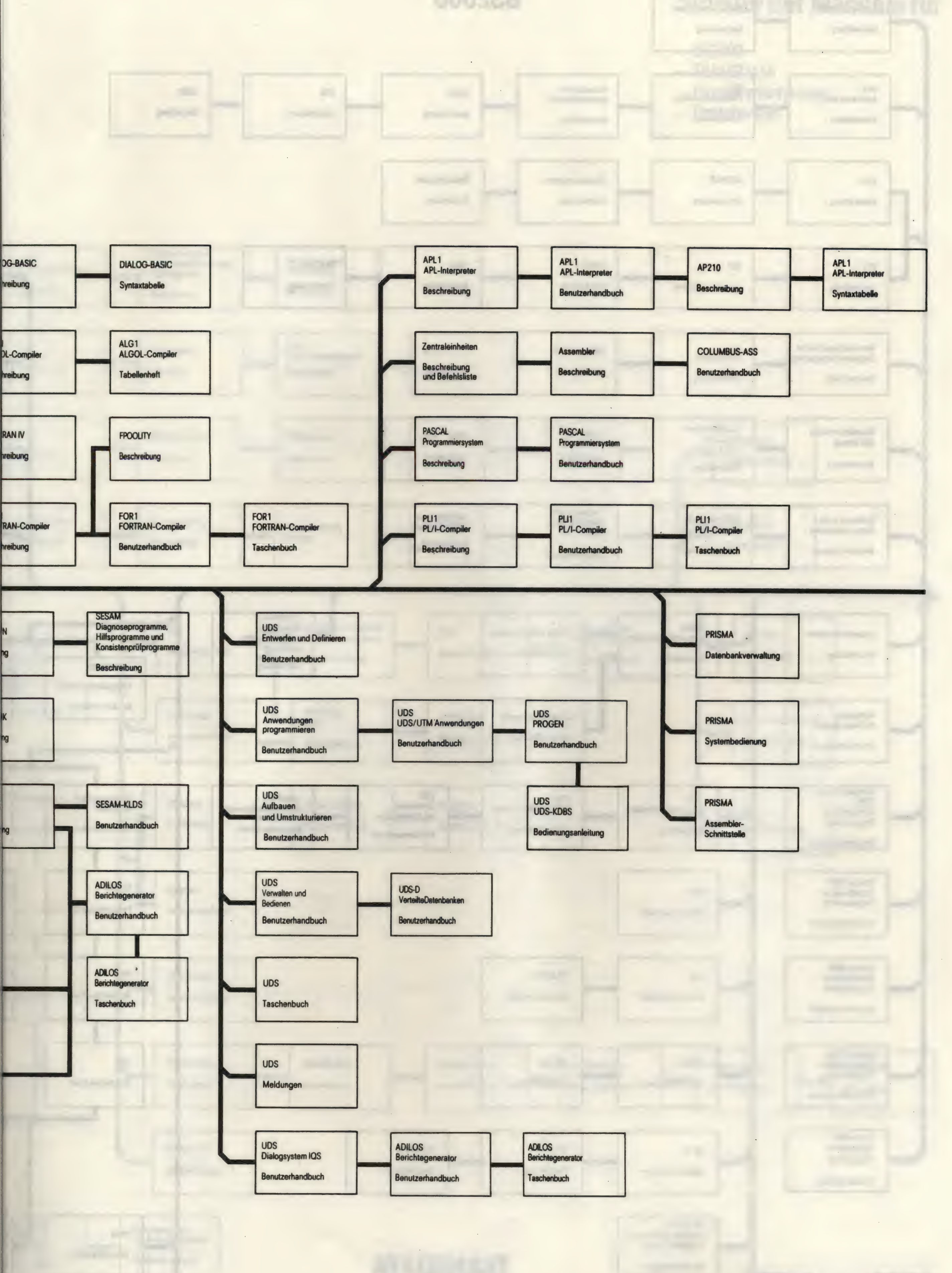


BS2000
Systemmeldungen Beschreibung
Systemkonventionen Beschreibung
DVS Bandverarbeitung Beschreibung
DVS Plattenverarbeitung Beschreibung
Makroaufrufe an den Ablaufteil Beschreibung
LEASY Beschreibung
RFA Beschreibung
ADAM Beschreibung
LMS Beschreibung
ARCHIVE Beschreibung
Dienstprogramme Beschreibung
Binder und Lader Beschreibung
EDT Beschreibung
EDT Anweisungsformate
EDOR Beschreibung
SORT Beschreibung
PERCON Beschreibung
Kommandosprache des Organisationsprogramms Beschreibung
JV Jobvariablen Beschreibung
Dialog-Testhilfe Beschreibung
Benutzerkommandos (SDF-Format) Beschreibung
Systemverwalter-kommandos (SDF-Format) Beschreibung
SDF-A Beschreibung
SDF-A Taschenbuch
Einführung in die Systemanwendung Benutzerhandbuch
Einführung in die Systembedienung Benutzerhandbuch
Einführung in die Dialogschnittstelle (SDF) Benutzerhandbuch
Einführung in die Geräteverwaltung Benutzerhandbuch
Systemkonventionen Beschreibung
Netzzugang für Datenstationen Benutzerhandbuch
Datensichtstation 9754 Schnittstelle für Programmierer Benutzerhandbuch
Drucker 9001 Schnittstelle für Programmierer Benutzerhandbuch
Drucker 9003 Schnittstelle für Programmierer Benutzerhandbuch
Drucker 9004 Schnittstelle für Programmierer Benutzerhandbuch
Drucker 9013 Schnittstelle für Programmierer Benutzerhandbuch
DCAM-Programmschnittstellen Beschreibung
DCAM-COBOL-Aufrufe Benutzerhandbuch
DCAM-Makroaufrufe Benutzerhandbuch
Anschluß an Datenpaket-Vermittlungsnetze (X.25, X.29) Beschreibung
Generierung von Vorzugskonfigurationen Benutzerhandbuch
UTM Programmschnittstellen Beschreibung
UTM KDCS-COBOL-Aufrufe Benutzerhandbuch
UTM Generierung und Administration Benutzerhandbuch
UTM-RPC1 UTM für RPG-Anwender Benutzerhandbuch
Netzmanagement im BS2000 Benutzerhandbuch
TIAM Benutzerhandbuch
NEOSP Benutzerhandbuch
FHS Benutzerhandbuch
IFG für FHS Benutzerhandbuch
TDA Benutzerhandbuch
ASMUS Benutzerhandbuch
ASMUS2 Benutzerhandbuch
ADS Benutzerhandbuch
RCOM Benutzerhandbuch
RCOM-S Beschreibung
RCOM-FTP Beschreibung
File Transfer in offenen Rechnernetzen FT-BS2000,FT-MSP Benutzerhandbuch
TRANSDATA
RBAS Benutzerhandbuch
Teletex-Anschluß an BS2000 mit TTX-B Benutzerhandbuch
OMNIS Benutzerhandbuch

Programmiersysteme



Datenbanksysteme



*in Vorbereitung

Stand 2.86

Inhalt

1	**Bereitstellung des Quellprogramms**	1-1
1.1	Einführung	1-1
1.1.1	Metasprache	1-1
1.1.2	Eingabemöglichkeiten für COBOL-Programme	1-1
1.2	Dateien	1-4
1.2.1	Übersicht über die Bereitstellung in Dateien	1-4
1.2.2	Eingabe in Dateien	1-4
1.2.3	Änderungen in Dateien	1-8
1.4	COBOL-Programme in LMS-Bibliotheken	1-19
1.4.1	Bibliotheksformate	1-19
1.4.1.1	Programmbibliotheken	1-19
1.4.1.2	Quellprogrammbibliotheken	1-19
1.4.1.3	Bindemodulbibliotheken	1-19
1.4.2	Zuweisen von LMS-Bibliotheken	1-19
1.4.3	Eingabe in LMS-Bibliotheken	1-20
1.4.4	Änderung in LMS-Bibliotheken	1-21
2	**Handhabung des COB1-Übersetzers**	2-1
2.1	Einführung	2-1
2.2	Eingabe	2-2
2.2.1	Übersicht über Eingabemöglichkeiten	2-2
2.2.2	Betriebsmittelzuweisung für die Eingabe	2-3
2.2.2.1	Eingabe über SYSDTA	2-4
2.2.2.2	Eingabe aus Bibliotheken	2-5
2.2.3	Eingabe von vollständigen Quellprogrammen	2-5
2.2.4	Eingabe von Quellprogrammteilen	2-10
2.2.5	Eingabe von Steueranweisungen (COBRUN-Operanden)	2-13
2.3	Ausgabe	2-17
2.3.1	Übersicht über Ausgabemöglichkeiten	2-17
2.3.2	Betriebsmittelzuweisung für die Ausgabe	2-18
2.3.3	Ausgabe von Bindemoduln (Objektmoduln)	2-22
2.3.4	Ausgabe von Listen	2-23
2.3.4.1	Steueranweisungsliste	2-25
2.3.4.2	Quellprogrammliste	2-26
2.3.4.3	Objektprogrammliste	2-31
2.3.4.4	Adreßliste	2-50
2.3.4.5	Fehlermeldungsliste	2-55
2.4	Ablauf	2-57
2.4.1	Übersicht über Steuerungsmöglichkeiten	2-57
2.4.2	PARAMETER-Kommando	2-59
2.4.3	COBRUN-Anweisung	2-62

3 Sicherstellung von Bindemoduln ... 3-1

4 Erzeugung ablauffähiger Programme ... 4-1
4.1 Einführung ... 4-1
4.2 Dynamisches Binden (DLL) ... 4-3
4.3 Statisches Binden (TSOSLNK) ... 4-6

5 Handhabung ablauffähiger Programme ... 5-1
5.1 Zuweisung von Betriebsmitteln ... 5-1
5.1.1 Übersicht ... 5-1
5.1.2 Bearbeitung von Systemdateien ... 5-1
5.1.3 Bearbeitung von Dateien ... 5-5
5.2 Programmaufruf und Programmbeendigung ... 5-8
5.3 Dialogtesthilfe AID ... 5-11
5.3.1 Einführung ... 5-11
5.3.2 Voraussetzungen für das symbolische Testen ... 5-12
5.3.3 Symbolisches Testen mit AID ... 5-13
5.4 Prozeßschalter und Benutzerschalter ... 5-16
5.5 Verwendung von Jobvariablen ... 5-19
5.6 Zugriff auf Übersetzer- und Betriebssysteminformationen ... 5-21

6 Programmierhinweise ... 6-1
6.1 Programmierbare Testhilfen ... 6-1
6.1.1 Testhilfezeilen ... 6-1
6.1.2 EXHIBIT-Anweisung ... 6-2
6.1.3 TRACE-Anweisung ... 6-5
6.2 Mehrfachbenutzbare Programme ... 6-7
6.2.1 Allgemeines ... 6-7
6.2.2 Segmentierung ... 6-7
6.2.3 Shared Code ... 6-10
6.2.4 Mehrfachbenutzbarkeit des Ablaufzeitsystems ... 6-12
6.3 Programmverknüpfungen ... 6-13
6.3.1 Programmverknüpfung COBOL-COBOL ... 6-13
6.3.2 Programmverknüpfung COBOL-ASSEMBLER (ASSEMBLER-COBOL) ... 6-16
6.4 Ein-Ausgabe von Daten über Systemdateien ... 6-21
6.5 Dateibearbeitung ... 6-23
6.5.1 Allgemeines ... 6-23
6.5.2 Dateiorganisationsformen, Zugriffsarten, Dateieröffnung, Übertragung der Daten ... 6-23
6.5.3 Sequentielle Dateiorganisation ... 6-27
6.5.4 Relative Dateiorganisation ... 6-37
6.5.5 Indizierte Dateiorganisation ... 6-44
6.5.6 Simultanverarbeitung für Dateien mit indizierter oder relativer Organisation 6-49
6.5.6.1 Indizierte Dateien ... 6-49
6.5.6.2 Relative Dateien ... 6-55
6.6 Sortieren und Mischen ... 6-57
6.7 Fixpunktausgabe und Wiederanlauf ... 6-60
6.7.1 Allgemeines ... 6-60
6.7.2 Fixpunktausgabe ... 6-60
6.7.3 Wiederanlauf ... 6-61

Anhang 1 Meldungen des COB1-Systems A-1
Anhang 2 Aufbau des COB1-Systems A-16
Anhang 3 Beschreibung des Objektmodulformats A-22
Anhang 4 Datenbankbedienung A-26
Literatur
Stichwörter



Bereitstellung des Quellprogramms 1

Handhabung des COB1-Übersetzers 2

Sicherstellung von Bindemoduln 3

Erzeugung ablauffähiger Programme 4

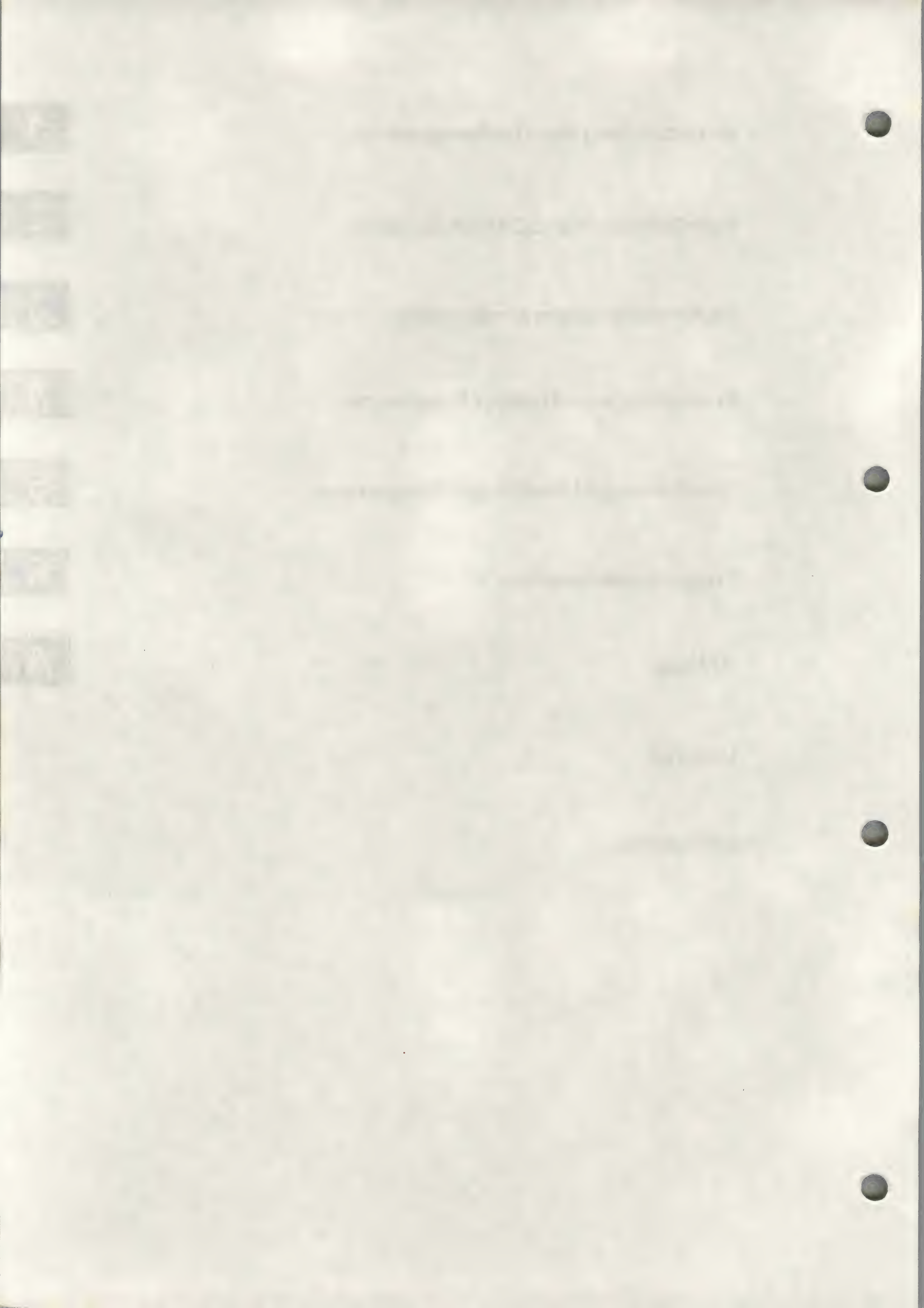
Handhabung ablauffähiger Programme 5

Programmierhinweise 6

Anhang A

Literatur

Stichwörter



1 Bereitstellung des Quellprogramms

1.1 Einführung

1.1.1 Metasprache für die BS2000-Anwendung

In diesem Benutzerhandbuch werden für die BS2000-Anwendung folgende metasprachliche Konventionen verwendet:

PARAM	Großbuchstaben bezeichnen Schlüsselwörter, die in dieser Form eingegeben werden müssen.
name	Kleinbuchstaben bezeichnen Variablen, die bei der Eingabe durch aktuelle Werte ersetzt werden müssen.
YES NO (unterstrichen)	Standardwerte sind unterstrichen. Sie werden vom Betriebssystem immer dann eingesetzt, wenn der Benutzer keine Angaben macht.
{YES NO (unterstrichen)}	Aus mehreren Möglichkeiten in einer geschweiften Klammer muß der Benutzer eine Angabe auswählen. Ist es ein Standardwert, so darf dieser weggelassen werden (kann aber auch angegeben werden).
{YES I NO}	Ein senkrechter Strich trennt verschiedene Angaben. Der Benutzer muß eine Angabe auswählen.
[ ]	Eckige Klammern schließen Wahlangaben ein, die der Benutzer auch weglassen darf.
( )	Runde Klammern müssen mit angegeben werden.
␣	Dieses Zeichen deutet an, daß mindestens ein Zwischenraum syntaktisch notwendig ist.
. . .	Drei Punkte bedeuten, daß die vor dem Komma stehende Einheit mehrmals hintereinander wiederholt werden kann.

Sonderzeichen sind ohne Veränderung zu übernehmen.

Hinweis: Für COBOL-Formatangaben gelten die üblichen COBOL-Konventionen (vgl. Manual „COB1 Beschreibung“ [1]).

1.1.2 Eingabemöglichkeiten für COBOL-Programme

Das vorliegende Kapitel vermittelt eine Übersicht über die Eingabemöglichkeiten eines COBOL-Quellprogramms in das Betriebssystem BS2000.

Nach seiner Codierung muß man ein COBOL-Programm für die Übersetzung bereitstellen. Bevor es jedoch der COB1-Übersetzer verarbeiten kann, muß sich der Benutzer entscheiden, auf welchem Weg er das Quellprogramm eingeben will:

- über **Floppy Disk** oder
- über **Lochkarten** oder
- im Dialogbetrieb an einer **Datenstation.**

Zusätzlich hat der Benutzer festzulegen, ob die Eingabe **direkt** in den COB1-Übersetzer erfolgen oder ob das Quellprogramm auf Magnetplatte zwischengespeichert, d.h. **indirekt** eingegeben werden soll (siehe Bild 1-1).

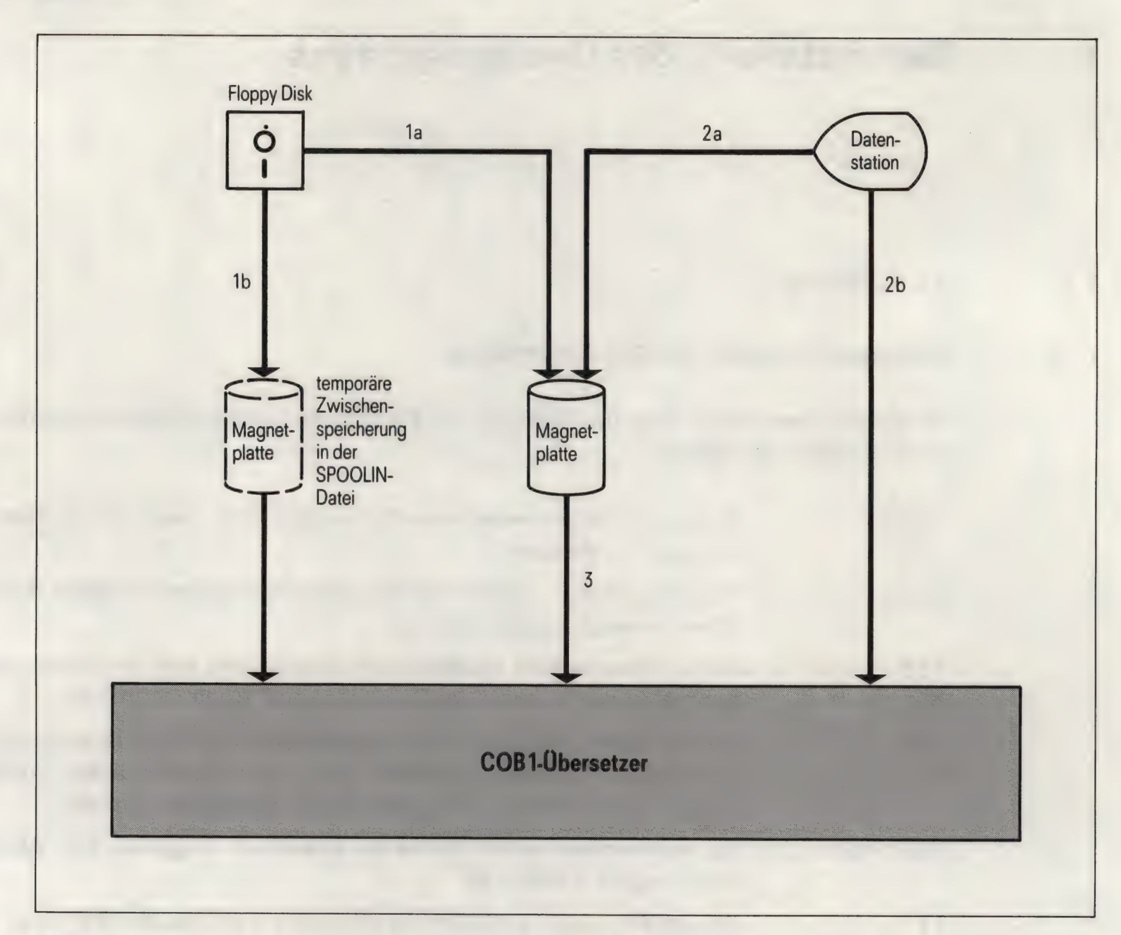


Bild 1-1

Direkte und indirekte Eingabe eines COBOL-Quellprogramms

In den Fällen ⓵a ⓶a wird das Quellprogramm zunächst auf Magnetplatte zwischengespeichert (siehe Kapitel 1.2 und 1.3), d. h. **indirekt** in den COB1-Übersetzer eingegeben. Bei ⓵b ⓶b ③ erfolgt die Eingabe **direkt** in den COB1-Übersetzer (siehe Kapitel 2.2).

Bei **indirekter** Eingabe des Quellprogramms steht das Programm für weitere Übersetzungsläufe auch nach der ersten Übersetzung bereit. Änderungen kann man mit Hilfe von Dienstprogrammen (z. B. mit einem Dateiaufbereiter) im Dialog mühelos durchführen.

Thema der nachfolgenden Abschnitte ist die Bereitstellung des Quellprogramms auf Magnetplatten. Daten auf Magnetplatten sind im BS2000 stets in sogenannten **Dateien** [4] organisiert und unter den Namen dieser Dateien abrufbar. Man unterscheidet je nach ihrer inneren Struktur zwei Dateitypen:

1. In einer einfachen **Datei** kann genau ein vollständiges Quellprogramm als Eingabe für den COB1-Übersetzer bereitstehen. Diesen Dateityp bezeichnet man auch als Quellprogramm-Datei.
2. In einer PLAM-Bibliothek oder einer LMS-Quellprogrammbibliothek können dagegen mehrere Quellprogramme oder auch Quellprogrammteile zur Eingabe in COB1 gespeichert werden.

Beispiel 1: Codiertes COBOL-Quellprogramm

Das folgende, einfache COBOL-Quellprogramm soll in den folgenden Abschnitten in einer Datei bzw. einer PLAM-Bibliothek oder LMS-Quellprogrammbibliothek zur Übersetzung bereitgestellt werden.

Die Bedeutung der einzelnen COBOL-Anweisungen geht aus der COB1-Beschreibung [1] hervor.

Siemens-Datenverarbeitung

COBOL-Ablochschema

Programm EINXEINS

Bearbeiter ______ Datum ______ Blatt

Lk-Nr. | A | B | Text | Kennung

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. EINXEINS.
000300*                  EINXEINS GIBT ZU EINER EINZUGEBENDEN
000400*                  ZAHL DAS EINMALEINS AUS.
000500*
000600 ENVIRONMENT DIVISION.
000700*
000800 DATA DIVISION.
000900 WORKING-STORAGE SECTION.
001000 77 ZAHL              PIC 99.
001100 77 ERGEBNIS          PIC ZZ9.
001200 77 I                 PIC 99.
001300 01 ERGEBNISTABELLE.
001400    05 ELEMENT OCCURS 10 TIMES PIC 999.
001500*
001600 PROCEDURE DIVISION.
001700 ANFANG.
001800     DISPLAY "ZWEISTELLIGE ZAHL EINGEBEN (ENDE BEI 0):"
001900         UPON TERMINAL.
002000     ACCEPT ZAHL FROM TERMINAL.
002100     IF ZAHL NOT NUMERIC
002200         DISPLAY "EINGABE MUSS NUMERISCH SEIN" UPON TERMINAL
002300         GO TO ANFANG.
002400     IF ZAHL = 0 STOP RUN.
002500     PERFORM RECHNEN VARYING I FROM 1 BY 1 UNTIL I > 10.
           GO TO ANFANG.
       RECHNEN.
           MULTIPLY I BY ZAHL GIVING ERGEBNIS, ELEMENT(I).
           DISPLAY I "*" ZAHL "=" ERGEBNIS UPON TERMINAL.
```

Bestell-Nr. U1855-J-Z58-1 68450. (0002,5)

1.2 Dateien

1.2.1 Übersicht über die Bereitstellung in Dateien

Abschnitt 1.2 zeigt, wie ein codiertes COBOL-Quellprogramm in eine Datei, die später der COB1-Übersetzer lesen soll, gebracht und dort geändert wird.

1.2.2 Eingabe in Dateien

Der COB1-Übersetzer kann sowohl die Zugriffsmethode SAM als auch ISAM benutzen, um ein Quellprogramm einzulesen. Zur Eingabe des Quellprogramms in Dateien dieser Typen stellt das BS2000 folgende Mittel zur Verfügung:

- das Kommando DATA [2] für Daten, die auf Floppy Disk gespeichert sind oder auf Lochkarten stehen. Das Kommando ist nur im Stapelbetrieb anwendbar.
- den Dateiaufbereiter EDT [5]; er ist dialogorientiert, aber auch im Stapelbetrieb anwendbar.

Liegt das Quellprogramm bereits auf Magnetplatte oder Magnetband gespeichert vor und soll es in eine bestimmte Datei übernommen werden, so bietet das BS2000 weitere Möglichkeiten:

Häufig genügt das COPY-Kommando [2] zum Kopieren von Dateien. Die Übernahme läßt sich aber auch mit Hilfe des Dateiaufbereiters EDT durchführen. Daneben gibt es Umsetzroutinen [3], die vor allem für Datenstrukturen eine Bedeutung haben, die vom Standardfall abweichen (z. B. Satzlängen größer als 256 Zeichen). Wegen der einfachen Struktur von Quelldaten wird hier auf diese Sonderfälle nicht näher eingegangen.

Das BS2000-Kommando DATA [2] zeigt an, daß ihm Datensätze im Kartenformat folgen, aus denen eine SAM- oder wahlweise eine ISAM-Datei auf gemeinschaftlichen Datenträgern erzeugt werden soll. Jedem Datensatz auf Floppy Disk bzw. Lochkarte entspricht dabei ein Datensatz in der erzeugten Datei.

Eine DATA-Datei legt das System während des Einspulvorgangs (SPOOL IN) an. Deshalb kann der Benutzer das Kommando nur in einspulenden Stapelprozessen und nicht in Dialog- oder ENTER-Prozessen verwenden.

Format des DATA-Kommandos

Operation	Operanden
DATA	dateiname [,[{SAM / ISAM}]] [,{NORMAL / MAXIMUM}]]

Die Angaben NORMAL bzw. MAXIMUM legen den Umfang der Speicherplatzzuweisung für die Datei „dateiname" mit 3 bzw. 12 PAM-Seiten fest.

Das erste DATA-Kommando eines Stapelauftrags muß unmittelbar dem LOGON-Kommando folgen. Die einzugebenden Daten werden durch das END-Kommando [2] abgeschlossen.

Für die Eingabe von weniger umfangreichen Quellprogrammen, Quellprogrammteilen und für Änderungen in bereits vorhandenen Quellprogrammen ist der Dialogbetrieb besonders geeignet.

Der **Dateiaufbereiter EDT** [5] bearbeitet SAM- und ISAM-Dateien, deren Sätze höchstens 256 Zeichen lang sind. Die ISAM-Datensätze müssen am Satzanfang einen numerischen Schlüssel besitzen, der bis zu 12 Zeichen lang sein darf. Der Benutzer kann mit dem EDT im Dialog- oder Stapelbetrieb arbeiten.

Beispiel 3: Eingabe eines COBOL-Quellprogramms mit dem EDT

(Das Protokoll wurde mit einer Schreibstation 8103 erstellt.)

```
/EXEC $EDT
% P5ØØ LOADING                                                              ①
*** EDITOR LOADED (VER11), BEGIN TYPE IN:
 1.         @TABS::$:8,12CHECK 72                                           ②

 1.         $IDENTIFICATION DIVISION.
 2.         $PROGRAM-ID. EINXEINS.
 3.         @P                                                              ③

 1.ØØØØ        IDENTIFICATION DIVISION.
 2.ØØØØ        PROGRAM-ID. EINXEINS.

 3.         $ENVIRONMENT DIVISION.
 4.         $DATA DIVISION.
 5.         $WORKING-STORAGE SECTION.
 6.         $77$ZAHL PIC 99.
 7.         $77$ERGEBNIS-PIC ZZ9.
 8.         $77$I PIC 99.
 9.         $PROCEDURE DIVISION.
1Ø.         $ANFANG.
11.         $$DISPLAY "ZWEISTELLIGE ZAHL EINGEGEBEN (ENDE BEI Ø):" UPON
            TERMINAL.
CHECK LINE LENGTH                                                           ④
12.         @P 11   1-72

11.ØØØØ        DISPLAY "ZWEISTELLIGE ZAHL EINGEBEN (ENDE BEI Ø):" UPON TERMI

12.         @ON 11 : 72-72 FIND 'I' DELETE SUFFIX                           ⑤

12.            -NAL.
13.         $$ACCEPT ZAHL FROM TERMINAL.
14.         $$IF ZAHL NOT NUMERIC
15.         $$DISPLAY "EINGABE ZAHL MUß NUMERISCH SEIN" UPON TERMINAL
16.         $$GO TO ANFANG.
17.         $$IF ZAHL = ZERO STOP RUN.
18.         $$PERFORM RECHNEN VARYING I FROM 1 BY 1 UNTIL I > 1Ø.
19.         $$GO TO ANFANG.
20.         $RECHNEN.
21.         $$MULTIPLY I BY ZAHL GIVING ERGEBNIS.
22.         $$DISPLAY I " * " ZAHL " = " ERGEBNIS UPON TERMINAL.
23.         @P&N                                                            ⑥

          IDENTIFICATION DIVISION.
          PROGRAM-ID. EINXEINS.
          ENVIRONMENT DIVISION.
          DATA DIVISION.
          WORKING-STORAGE SECTION.
.......................................................................
          RECHNEN.
              MULTIPLY I BY ZAHL GIVING ERGEBNIS.
              DISPLAY I " * " ZAHL " = " ERGEBNIS UPON TERMINAL.
23.         @WRITE 'QUELL.EDT'                                              ⑦
23.         @HALT                                                           ⑧
/
```

① Aufruf des Programms EDT.

② Die Anweisung @TABS... vereinbart $ als Tabulatorzeichen, das auf die Spalten 8 und 12 ausrichten soll. Außerdem soll der EDT melden, wenn die eingegebene Zeichenfolge länger als 72 Zeichen ist.

③ Nachdem die ersten beiden Datensätze des Quellprogramms in den EDT-Speicherbereich eingegeben wurden, kann man sie mit der Anweisung @P ausgeben lassen, um die Funktion des Tabulators zu überprüfen. Die Ausgabe zeigt auch die Zeilenzahlen der Sätze, die sie im EDT-Speicherbereich erhalten.

④ Diese EDT-Meldung weist darauf hin, daß die Eingabe länger als 72 Zeichen ist.

⑤ Auf Zeile 11 sollen, falls in Spalte 72 der Buchstabe I steht, die nachfolgenden Zeilen gelöscht werden (Beispiele für weitere Korrekturen in 1.2.3).

⑥ Alle Sätze im EDT-Speicherbereich sind ohne ihre Zeilenzahlen auszugeben.

⑦ Diese @WRITE-Anweisung schreibt den Inhalt des EDT-Speicherbereichs in eine SAM-Datei namens QUELL.EDT.

⑧ Ende des Programms EDT.

Leerseite durch den Nachtrag vom August 1986

1.2.3 Änderungen in Dateien

Aus unterschiedlichen Gründen kann man Quelldaten in einer Datei ändern wollen, zum Beispiel wegen

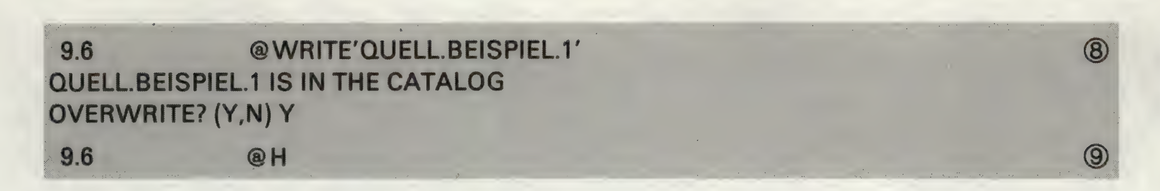
- Tippfehlern;
- formalen Fehlern, die der Übersetzer meldet;
- logischen Fehlern, die bei einem Testlauf des ablauffähigen Programms festgestellt werden.

Im folgenden wird gezeigt, wie sich derartige ÄnderungenimDialog mit Hilfe von EDT durchführen lassen.

Beispiel 5: Änderungen eines fehlerhaften Programms mit EDT

Das Protokoll wurde an einer Schreibstation 8103 erstellt und in der Datei QUELL.BEISPIEL.1 abgespeichert.

```
/EXEC $EDT
% P500 LOADING
 PROGRAM EDT/12.1 STARTED
 1.         @READ 'QUELL.BEISPIEL.1'                                    ①
98.         @PRINT 1-12                                                 ②
 1.0000              ID DIVISION.
 2.0000              PROGRAM-ID,    BEISPIEL.
 3.0000              ENVIRONMENT DIVISION.
 4.0000              ENVIRONMENT DIVISION.
 5.0000              INPUT-OUTPUT SECTION.
 6.0000              FILE-CONTROL.
 7.0000                     SELECT SORTDAT ASSIGN TO DISC.
 8.0000                     SELECT PRINTDAT ASSIGN TO SYSLST RESERVE NO.
 9.0000                     SELECT SAMDAT ASSIGN TO DA-590-S-SYS010.
10.0000              FILE SECTION.
11.0000              SD SORTDAT
12.0000                     RECORDING F
98.         @DELETE 3                                                   ③
98.         @TABS::[: 8,12                                              ④
98.         @SET 9.5 :[DATA DIVISION.
 9.6        @CHECK ON                                                   ⑤
 9.6        @ON 2 CHANGE ',' TO'.'                                      ⑥
 2.0000              PROGRAM-ID.    BEISPIEL
 9.6                 @PRINT 1-12 N                                      ⑦
            ID DIVISION.
            PROGRAM-ID.    BEISPIEL.
            ENVIRONMENT DIVISION.
            INPUT-OUTPUT SECTION.
            FILE-CONTROL.
                     SELECT SORTDAT ASSIGN TO DISC.
                     SELECT PRINTDAT ASSIGN TO SYSLST RESERVE NO.
                     SELECT SAMDAT ASSIGN TO DA-590-S-SYS010.
            DATA DIVISION.
            FILE SECTION.
            SD SORTDAT
                     RECORDING F
```



```
 9.6            @WRITE'QUELL.BEISPIEL.1'                        ⑧
QUELL.BEISPIEL.1 IS IN THE CATALOG
OVERWRITE? (Y,N) Y
 9.6            @H                                              ⑨
```

① Die Datei QUELL.BEISPIEL.1 wird in den EDT-Speicherbereich gelesen.

② Die Sätze 1 bis 12 werden mit EDT-Satznummern aus dem EDT-Speicherbereich auf die Datenstation ausgegeben.

③ Satz 3 wird gelöscht.

④ Der Tabulator wird mit der @TABS-Anweisung gesetzt. Er wirkt bei der nachfolgenden Eingabe (@SET) eines neuen Satzes, der die EDT-Satznummer 9.5 erhält.

⑤ Damit wird erreicht, daß im folgenden veränderte Datensätze auf der Datenstation protokolliert werden.

⑥ In Satz 2 wird das erste Komma (,) in einen Punkt (.) geändert. Da der CHECK-Modus eingeschaltet wurde, protokolliert der EDT diese Änderungen automatisch.

⑦ Zur Kontrolle werden die Sätze 1 bis 12 erneut ausgegeben, diesmal ohne die EDT-Satznummern.

⑧ Der Inhalt des EDT-Speicherbereichs soll in die Datei QUELL.BEISPIEL.1 zurückgeschrieben werden. Daraufhin meldet der EDT, daß diese Datei bereits im Katalog eingetragen ist und fragt, ob ihr Inhalt überschrieben werden soll. Die Antwort Y hat dies zur Folge.

⑨ Der EDT wird beendet und in den Systemmodus übergegangen.

Leerseite durch den Nachtrag vom August 1986

1.4 COBOL-Programme in LMS-Bibliotheken

Das Bibliotheksprogramm LMS [21] erstellt und verwaltet Bibliotheken und bearbeitet Bibliothekselemente auf Magnetplatte und Magnetband.

LMS kann auch mit COBLUR erstellte Bibliotheken lesen, ändern jedoch nicht (siehe „Umstellung von . . . COBLUR auf LMS“ in [21]).

1.4.1 Bibliotheksformate

Von den Bibliotheken, die LMS bearbeiten kann (genaue Beschreibung siehe unter „Bibliotheksformate“ und „Elemente“ in [21]), sind die folgenden für COBOL-Programme von Bedeutung:

- Programmbibliotheken zum Speichern von Quellprogrammen, Moduln, Listen, Prozeduren usw.
  Zugriffsmethode: PLAM
- Quellprogrammbibliotheken zum Speichern von Quellprogrammen, Quellprogrammteilen, Listen, Prozeduren
  Zugriffsmethode: ISAM
- Bindemodulbibliotheken zum Speichern von Bindemoduln
  Zugriffsmethode: PAM.

1.4.1.1 Programmbibliotheken (PLAM-Bibliotheken)

Programmbibliotheken sind PAM-Dateien, die mit der Bibliothekszugriffsmethode PLAM bearbeitet werden. Sie bieten gegenüber anderen Bibliotheksformaten u. a. folgende Vorteile:

- Alle Bibliothekselementtypen können in einer Bibliothek abgelegt werden.
- Es dürfen gleichnamige Elemente existieren, die sich nur durch die Typ- oder Versionsbezeichnung unterscheiden müssen.

1.4.1.2 Quellprogrammbibliotheken

Quellprogrammbibliotheken sind ISAM-Dateien (KEYPOS=5, KEYLEN=8), die nur den Bibliothekselementtyp S enthalten können. In ihnen können sowohl vollständige Quellprogramme als auch Quellprogrammteile (COPY-Elemente) gespeichert werden.

1.4.1.3 Bindemodulbibliotheken

Bindemodulbibliotheken sind PAM-Dateien. Sie nehmen die vom Sprachübersetzer erzeugten Bindemoduln als Bibliothekselement mit dem Typ R auf (siehe „Sicherstellung von Bindemoduln“).

1.4.2 Zuweisen von LMS-Bibliotheken

Die Zuweisung von LMS-Bibliotheken (siehe auch „Zuweisen von Bibliotheken“ in [21]), die als Eingabe- und/oder Ausgabebibliothek für ein Bibliothekselement dienen sollen oder in denen ein Element bearbeitet werden soll, erfolgt mit der LIB-Anweisung (siehe diese in [21]). Diese bestimmt u. a., ob es sich um eine neu einzurichtende oder eine bereits vorhandene Bibliothek handelt und welches Bibliotheksformat sie haben soll oder hat.

Soll die zu bearbeitende Bibliothek in LMS-Anweisungen über die Bibliothekskurzbezeichnung oder den Dateikettungsnamen angesprochen werden, muß vor dem LMS-Aufruf oder spätestens vor der ersten LMS-Anweisung ein /FILE-Kommando gegeben werden.

Format des /FILE-Kommandos:

```
/FILE bibliotheksname, {LINK=dateikettungsname}
                       {LINK=LIBlib           }
```

bibliotheksname	Name der LMS-Bibliothek
dateikettungsname	Name, mit dem die Bibliothek in LMS-Anweisungen angesprochen werden kann (wird in die Prozeßdateitabelle (TFT) eingetragen)
LIBlib	Dateikettungsname mit Bibliothekskurzbezeichnung LIB fester Bestandteil lib Kurzbezeichnung der Bibliothek; Ganzzahl, die immer dreistellig anzugeben ist

Soll in der LMS-Anweisung die Bibliothekskurzbezeichnung verwendet werden, ist als Dateikettungsname LIBlib zu verwenden. Die Kurzbezeichnung ist in runde Klammern zu setzen; führende Nullen dürfen weggelassen werden.

1.4.3 Eingabe in LMS-Bibliotheken

In LMS-Bibliotheken können Quellprogramme eingegeben werden

- aus Dateien
- aus einer Bibliothek
- über die Systemdatei SYSDTA bzw. SYSIPT, d. h. von einer Datenstation oder einer temporären SPOOLIN-Datei.

Eingabebibliotheken müssen vor dem LMS-Aufruf durch ein /FILE-Kommando zugewiesen werden, wenn sie über den Dateikettungsnamen oder die Kurzbezeichnung angesprochen werden sollen. Die Datenstation bzw. die temporäre SPOOLIN-Datei muß über das SYSFILE-Kommando zugewiesen werden.

Hinweis: Soll ein COBOL-Quellprogramm mit LMS aus einer katalogisierten ISAM-Datei in eine PLAM-Bibliothek aufgenommen werden, so darf dabei nicht der Verarbeitungsoperand PAR KEY = YES wirksamsein. Andernfalls kann COB1 dieses Quellprogrammelement nicht bearbeiten.

Beispiel 9a: Übernahme einer katalogisierten Datei in eine LMS-Bibliothek

```
/FILE LMS.SOURCE,LINK=LIB001                    ①
/EXEC $LMS                                      ②
$LIB LID=(001),STATE=NEW,USAGE=OUT              ③
$ADDS COB.PROG>COBELEM1                         ④
$END                                            ⑤
```

① Die Bibliothek LMS.SOURCE wird zugewiesen und mit dem Dateikettungsnamen LIB001 verknüpft.

② Aufruf von LMS

③ LMS.SOURCE mit der Kurzbezeichnung 1 (LID=001) wird als neueinzurichtende (STATE=NEW) Ausgabebibliothek (USAGE=OUT) erklärt. Da der Operand FORMAT nicht angegeben ist, wird sie standardmäßig als Programmbibliothek (Typ PL) eingerichtet.

④ Die katalogisierte Datei COB.PROG wird als Element vom Typ S unter dem Namen COBELEM1 in die LMS-Bibliothek aufgenommen.

⑤ Beendigung des Programms LMS, Schließung aller geöffneten Bibliotheken und Dateien.

Beispiel 9b: Übernahme eines COBLUR-Elementes in eine LMS-Bibliothek

```
/FILE COBLUR.PR, LINK=LIB012                              ①
/FILE LMS.SOURCE, LINK=LIB013                             ②
/EXEC $LMS                                                ③
$LIB LID=(13),USAGE=OUT,FORMAT=PL,STATE=OLD               ④
$LIB LID=(12),USAGE=IN                                    ⑤
$DUPS COBLUREL(12)>LMSELE4                                ⑥
$END                                                      ⑦
```

① Die COBLUR-Bibliothek COBLUR.PR wird zugewiesen und mit dem Dateikettungsnamen LIB012 verknüpft.

② Die LMS-Bibliothek LMS.SOURCE wird zugewiesen und mit dem Dateikettungsnamen LIB013 verknüpft.

③ Aufruf von LMS

④ LMS.SOURCE mit der Kurzbezeichnung 13 (LID=(13)), die bereits existiert (STATE=OLD) als Programmbibliothek (FORMAT=PL), wird als Ausgabebibliothek erklärt (USAGE=OUT).

⑤ COBLUR.PR mit der Kurzbezeichnung 12 (LID=(12)), die bereits existiert, wird als Eingabebibliothek erklärt (USAGE=IN).

⑥ Das COBLUR-Element COBLUREL aus der Bibliothek COBLUR.PR(12) wird dupliziert und unter dem neuen Namen LMSELE4 in LMS.SOURCE eingetragen.
Enthält die COBLUR-Bibliothek in mehreren Bibliotheksabschnitten Elemente gleichen Namens, so wird nur das zuerst gefundene Element übernommen.

⑦ Beendigung des Programms LMS und Schließen aller geöffneten Bibliotheken.

Kopieren von vollständigen Bibliotheken ist mit COPS * durchzuführen, sofern nicht mehrere Elemente gleichen Namens vorhanden sind (siehe LMS Beschreibung [21], Anhang).

Sind in einer COBLUR-Bibliothek in verschiedenen Bibliotheksabschnitten Elemente gleichen Namens, die mit COPS * kopiert werden sollen, so empfiehlt sich:

PAR OVERWRITE=NO (Standard)

Dadurch wird nur das zuerst gefundene Element mit dem jeweiligen Namen übernommen.

PAR OVERWRITE=YES

Dadurch wird das zuletzt gefundene Element übernommen.

Nachdem ein Element kopiert wurde, muß es in der COBLUR-Bibliothek gelöscht werden, damit das nächste Element gleichen Namens durch

DUPS elementnamealt>elementnameneu

umbenannt und in die LMS-Bibliothek übernommen werden kann.

1.4.4 Änderungen in LMS-Bibliotheken

Mit Hilfe eines Editors (EDT, EDOR) kann ein Quellprogramm in einer LMS-Bibliothek geändert werden; nötig ist die LMS-Anweisung EDTS bzw. EDRS. — Die Anweisung CORS verändert ohne Editor-Aufruf. Nötig sind dann ergänzende Anweisungen, z. B. *INSERT (Einfügen), *DELETE (Löschen), *REPLACE (Ersetzen).

2 Handhabung des COB1-Übersetzers

2.1 Einführung

In Kapitel 1 wird gezeigt, wie der Benutzer sein COBOL-Quellprogramm in einer Datei bzw. einer COBOL-Bibliothek bereitstellen kann. Anschließend daran muß das Quellprogramm in eine Anweisungsfolge umgesetzt werden, die der Rechner verarbeiten kann. Diese Umwandlung eines COBOL-Quellprogramms in Maschinenbefehle läßt sich im BS2000 in zwei Schritten erreichen:

1. **Übersetzen:** Der COB1-Übersetzer erzeugt als Vorstufe des ablauffähigen Programms einen Objektmodul — und zwar einen sogenannten **Bindemodul** — und Protokolle. Der Benutzer steuert

   — Eingabe
   — Ausgabe
   — Ablauf

   des COB1-Übersetzers. Dies wird im Kapitel 2 beschrieben.
2. **Binden:** Der Bindemodul bzw. mehrere unabhängig voneinander erzeugte Bindemoduln wird bzw. werden zusammen mit Ablaufzeitsystem-Moduln zusammengefügt (= gebunden). Es entsteht ein sogenannter **Lademodul**; siehe Kapitel 4.

Das Ergebnis von Übersetzungs- und Binderlauf, das gebundene Programm (Lademodul), ist dann auf der Datenverarbeitungsanlage ablauffähig, d. h. die Folge seiner Maschinenbefehle wird vom Rechner ausgeführt.

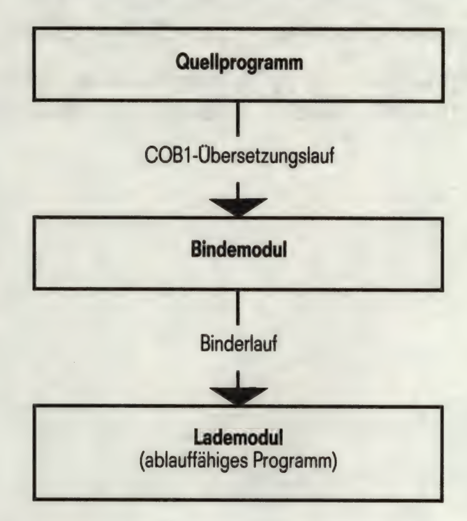


Bild 2-1

Der Weg zum ablauffähigen Programm

2.2 Eingabe

2.2.1 Übersicht über Eingabemöglichkeiten

Eingaben in den COB1-Übersetzer können folgende Quelldaten sein:

— **Quellprogramm**: Dies ist die einfachste Eingabemöglichkeit.

— **Quellprogrammteile**: Sie können zusätzlich in ein Quellprogramm eingebaut werden.

— **Steueranweisungen**: Sie steuern Übersetzungsablauf und Übersetzerausgabe.

Die Quelldaten können über unterschiedliche physikalische Datenträger den Übersetzer erreichen, nämlich Datenstation, Floppy Disk, Lochkarten, Magnetplatten, Magnetbänder. Logisch gesehen vereinheitlicht sich jedoch das Bild auf zwei **Eingabemedien:**

Eingabedaten	Eingabe ist möglich ... von SYSDTA	... aus einer COBOL-Bibliothek	Beispiel in Abschnitt
vollständiges Quellprogramm	X	X	2.2.3
Quellprogrammteile	–	X	2.2.4
Steueranweisungen	X	–	2.2.5

Der Übersetzer erwartet seine Eingabedaten zunächst stets von der System-Eingabedatei SYSDTA. Aus dieser **„Quelleingabe"** kann sich ergeben, daß zusätzlich Quelldaten aus einer COBOL-Bibliothek einzulesen sind, ein Vorgang, den man als **Sekundäreingabe** bezeichnet.

Bild 2-2 gibt eine Übersicht über die Eingabemöglichkeiten in den COB1-Übersetzer.

Zusätzlich zu der Unterscheidung von Eingaben aus SYSDTA oder aus einer COBOL-Bibliothek lassen sich die Eingabeformen in den Übersetzer auch in **direkte** und **indirekte Eingabe** aufgliedern (siehe Kapitel 1).

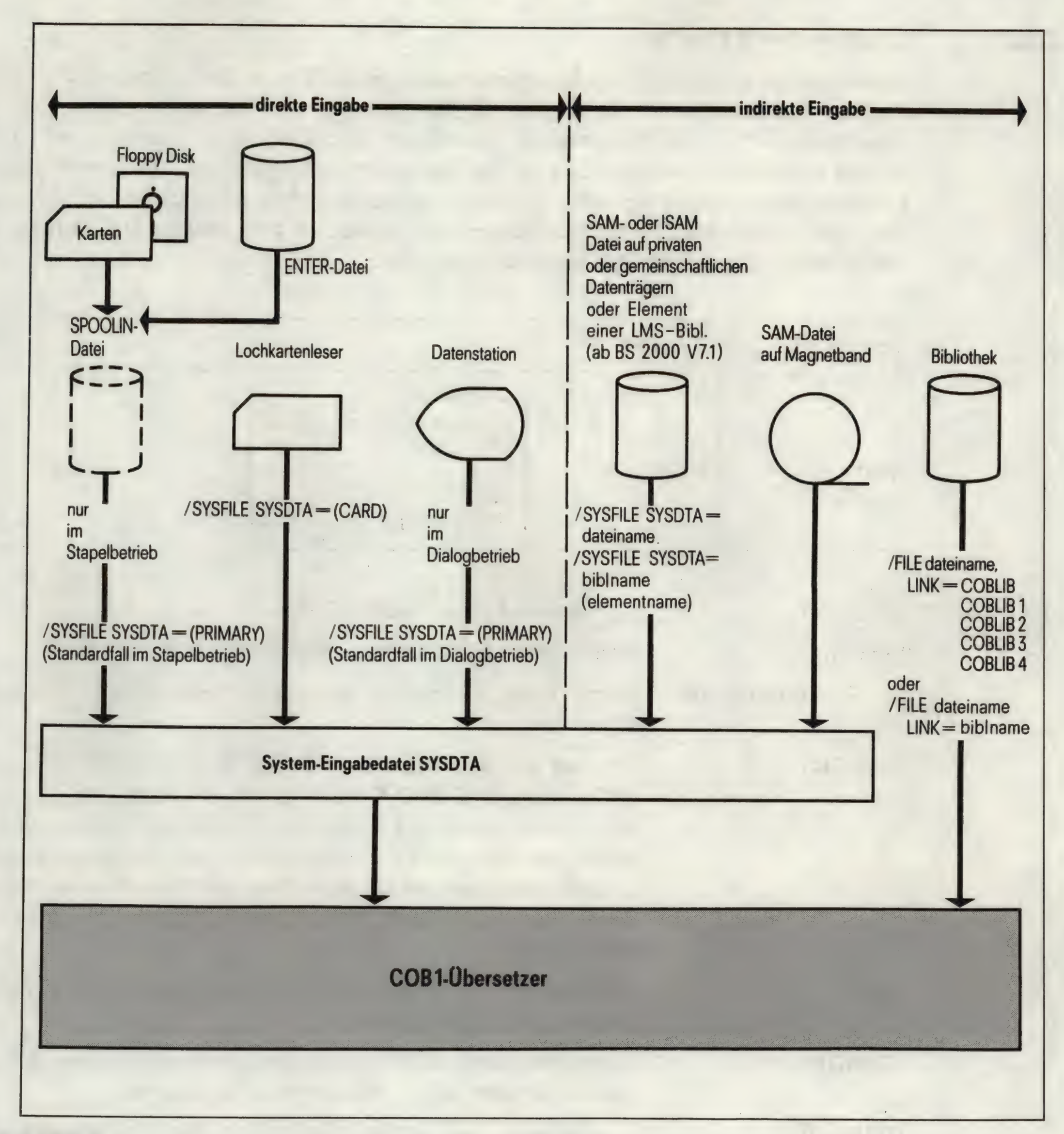


Bild 2-2

Eingabeweg für den COB1-Übersetzer

Die Zuweisung der Betriebsmittel erfolgt durch die angegebenen BS2000-Kommandos.

2.2.2 Betriebsmittelzuweisung für die Eingabe

Datenstation, Lochkartenleser, Plattendateien zählen für den COB1-Übersetzer zu den Betriebsmitteln der Eingabe (siehe Abbildung 2-2). Der Benutzer hat die Auswahl der jeweils erforderlichen Betriebsmittel vor dem Aufruf des Übersetzers zu treffen, und zwar

- für die Eingabe von SYSDTA mit dem SYSFILE-Kommando,
- für die Eingabe aus einer COBOL-Bibliothek mit dem FILE-Kommando des BS2000.

2.2.2.1 Eingabe über SYSDTA

Standardmäßig ist der **Eingabe-Systemdatei SYSDTA** im Dialogbetrieb die Datenstation, im Stapelbetrieb die SPOOLIN- bzw. ENTER-Datei [2] des Prozesses zugewiesen. Sollen die Eingabedaten für die Übersetzung aus diesen Eingabequellen kommen, so ist kein Kommando erforderlich. Andernfalls hat der Benutzer die Möglichkeit, die Systemdatei SYSDTA mit Hilfe des SYSFILE-Kommandos einer Datei, einer Prozedurdatei, einem Lochkartenleser oder einem Element in einer Quellprogramm-Bibliothek zuzuweisen. Das Format für den betreffenden Teil des SYSFILE-Kommandos lautet:

Operation	Operand
SYSFILE	SYSDTA = { dateiname biblname (elementname) (SYSCMD) (mn) (CARD) (PRIMARY) }

dateiname	Name einer katalogisierten Datei
biblname	Name einer LMS-Quellprogrammbibliothek
(elementname)	Name eines Elementes der LMS-Quellprogrammbibliothek (max. 8 Zeichen)
(SYSCMD)	schaltet die Systemdateien SYSDTA und SYSCMD zusammen, so daß aus der SYSCMD-Datei nicht nur BS2000-Kommandos, sondern auch Daten kommen können. Bei Prozeßbeginn sind beide standardmäßig insofern zusammengeschaltet, als sie beide standardmäßig im Dialogbetrieb der Datenstation, im Stapelbetrieb der SPOOLIN-Datei zugewiesen sind. Diese Zuordnung wird als Primärzuweisung bezeichnet.
(mn)	Kartenleser bzw. Disketten-Gerät mit einem 2 Byte langen mnemotechnischen Gerätenamen.
(CARD)	Zuordnung eines Kartenlesers und Aufforderung an den Operateur, einen Kartenstapel in den Kartenleser einzulegen.
(PRIMARY)	Zurückzuweisung von SYSDTA auf die Primärzuweisung (siehe SYSCMD).

Beispiele für die Eingabe von SYSDTA in den COB1-Übersetzer siehe in den Abschnitten 2.2.3 bis 2.2.5.

2.2.2.2 Eingabe aus Bibliotheken

Der COB1-Übersetzer kann vollständige Quellprogramme oder Quellprogrammteile aus einer Bibliothek lesen. Es gibt mehrere Möglichkeiten die Bibliothek oder gegebenenfalls die Bibliotheken zuzuweisen.

- Einlesen ganzer Quellprogramme

  SYSDTA-Zuweisung

  FILE bibliothekname,LINK = SRCLIB
  in Verbindung mit
  COBRUN SRCELEM = elementname

  SYSDTA-Umweisung über die Cobrun-Anweisung END

- Einlesen von Quellprogrammteilen

  FILE-Zuweisung über Standard-LINK-Namen
  COBLIB und
  COBLIB1 — COBLIB4 (COPY-Hierarchie)

  FILE-Zuweisung über den LINK-Namen aus der COPY-Anweisung

2.2.3 Eingabe von vollständigen Quellprogrammen

Vollständige Quellprogramme können eingegeben werden (siehe auch Bild 2-2)

- aus der SPOOLIN-Datei (nur Stapelbetrieb)
- vom Lochkartenleser
- von der Datenstation (nur Dialogbetrieb)
- aus einer Quellprogrammdatei
- aus einer Bibliothek

Zu den einzelnen Eingabeformen siehe die folgenden Beispiele.

Beispiel 10: Direkte Eingabe eines Quellprogramms über die SPOOLIN-Datei

Die Folge der benötigten Kommandos sowie das Quellprogramm können

- sich auf Lochkarten befinden*)
- auf Floppy Disk gespeichert sein*)
- in einer Datei stehen, die mit einem ENTER-Kommando [2] aufgerufen werden muß (ENTER-Datei).

---

*) Das System übernimmt den gesamten Inhalt der Lochkarten bzw. der Daten auf Floppy Disk in eine SPOOLIN-Datei namens S.INtsn. Außerdem wird der künftige Stapelprozeß in die Auftragswarteschlange eingetragen.

Jeder Zeile entspricht eine Lochkarte bzw. ein Datensatz.

```
/LOGON...              ①
/EXEC $COB1            ②
Quellprogramm          ③
/LOGOFF                ④
```

① Das LOGON-Kommando identifiziert den Benutzer gegenüber dem System. Zwischen LOGON- und EXECUTE-Kommando können weitere Kommandos erforderlich sein, wie zum Beispiel das PARAMETER-Kommando zur Steuerung des COB1-Übersetzers.

② Das EXECUTE-Kommando lädt und startet den COB1-Übersetzer. Das Zeichen $ vor dem Namen des Übersetzers veranlaßt das System, den Übersetzer unter der Kennung des Systems, TSOS, zu suchen.

③ Das Quellprogramm — abgeschlossen mit /* — folgt dem EXEC-Kommando und reicht bis zum nächsten Kommando, das mit einem Schrägstrich in Spalte 1 beginnt.

④ LOGOFF beendet den Prozeß. Vor diesem Kommando können beliebige weitere Kommandos stehen, beispielsweise kann ein Binderlauf erfolgen.

Beispiel 11: Direkte Eingabe eines Quellprogramms im Dialog (über Lochkartenleser)

Ausschnitt aus einem Dialogprozeß:

```
/TYPE BITTE LK FUER MEIER IN LK-LESER    ①
/SYSFILE SYSDTA=(CARD)                   ②
/EXEC $COB1                              ③
/SYSFILE SYSDTA=(PRIMARY)                ④
```

① Das TYPE-Kommando meldet dem Operateur, welche Lochkarten im lokalen Lochkartenleser verarbeitet werden sollen.

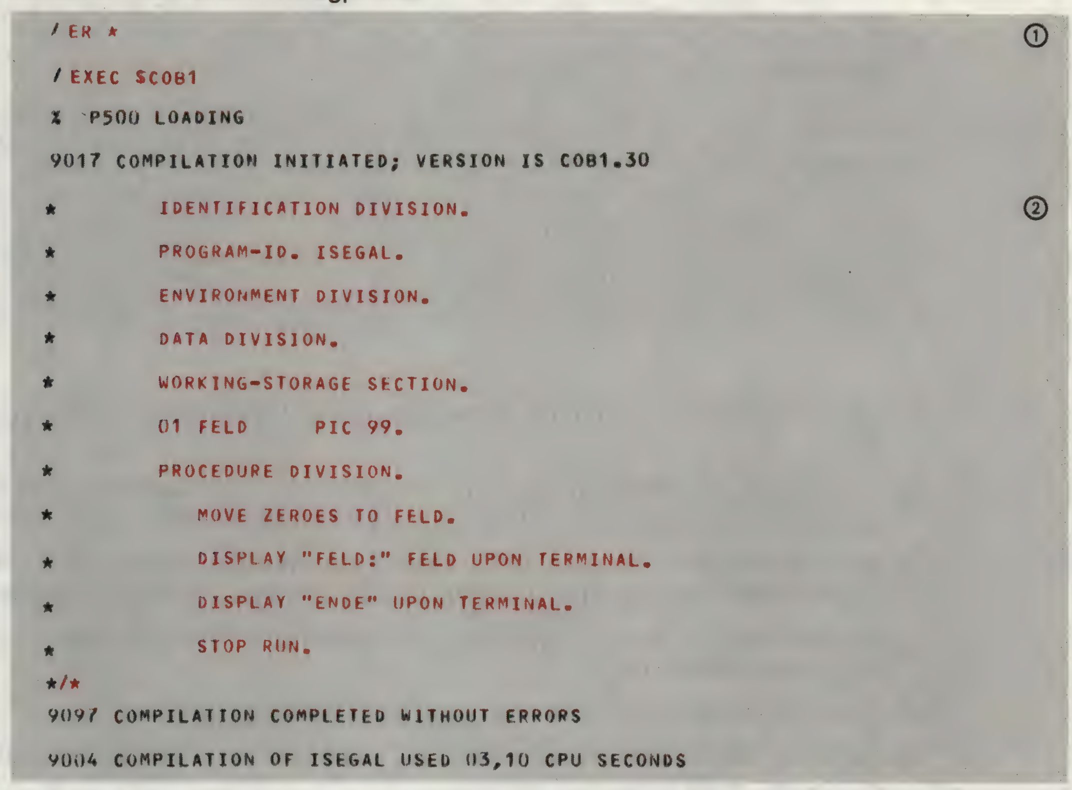
② Die Systemdatei SYSDTA ist im Dialogbetrieb standardmäßig der Datenstation zugeordnet. Da das Quellprogramm sich nun im Kartenleser befindet, wird SYSDTA auf dieses Gerät gelegt.

③ Der Übersetzer wird aufgerufen und liest das Quellprogramm von SYSDTA, d. h. dem Lochkartenleser.

④ Die Systemdatei SYSDTA wird wieder der Datenstation zugeordnet und damit der Kartenleser freigegeben.

Beispiel 12: Direkte Eingabe eines Quellprogramms im Dialog (über Datenstation)

Ausschnitt aus einem Dialogprozeß:



```
/ ER *                                                                  ①
/EXEC $COB1
% P500 LOADING
9017 COMPILATION INITIATED; VERSION IS COB1.30
*       IDENTIFICATION DIVISION.                                        ②
*       PROGRAM-ID. ISEGAL.
*       ENVIRONMENT DIVISION.
*       DATA DIVISION.
*       WORKING-STORAGE SECTION.
*       01 FELD     PIC 99.
*       PROCEDURE DIVISION.
*           MOVE ZEROES TO FELD.
*           DISPLAY "FELD:" FELD UPON TERMINAL.
*           DISPLAY "ENDE" UPON TERMINAL.
*           STOP RUN.
*/*
9097 COMPILATION COMPLETED WITHOUT ERRORS
9004 COMPILATION OF ISEGAL USED 03,10 CPU SECONDS
```

① Der COB1-Übersetzer wird aufgerufen und erwartet die Eingabe von der Systemdatei SYSDTA, in diesem Fall also der Datenstation.

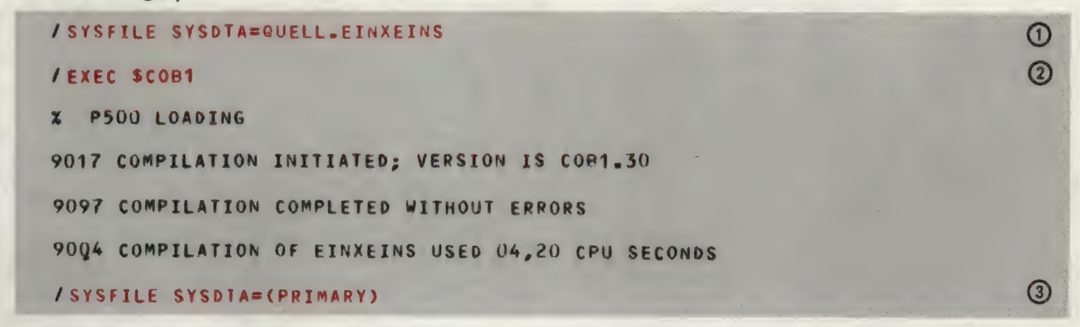
② Den Stern (*) gibt der Übersetzer als Aufforderung zur Eingabe des Quellprogramms aus.

③ /* beendet die Eingabe.

Diese Eingabeform hat wenig praktische Bedeutung, da nach der Übertragung einer Eingabe Fehler nicht mehr korrigiert werden können.

Beispiel 13: Indirekte Eingabe aus einer Quellprogramm-Datei

(Die Kommandofolge gilt für Dialog- und Stapelbetrieb, das Beispiel zeigt den Ausschnitt eines Dialogs.)



```
/SYSFILE SYSDTA=QUELL.EINXEINS                                          ①
/EXEC $COB1                                                             ②
% P500 LOADING
9017 COMPILATION INITIATED; VERSION IS COB1.30
9097 COMPILATION COMPLETED WITHOUT ERRORS
9004 COMPILATION OF EINXEINS USED 04,20 CPU SECONDS
/SYSFILE SYSDTA=(PRIMARY)                                               ③
```

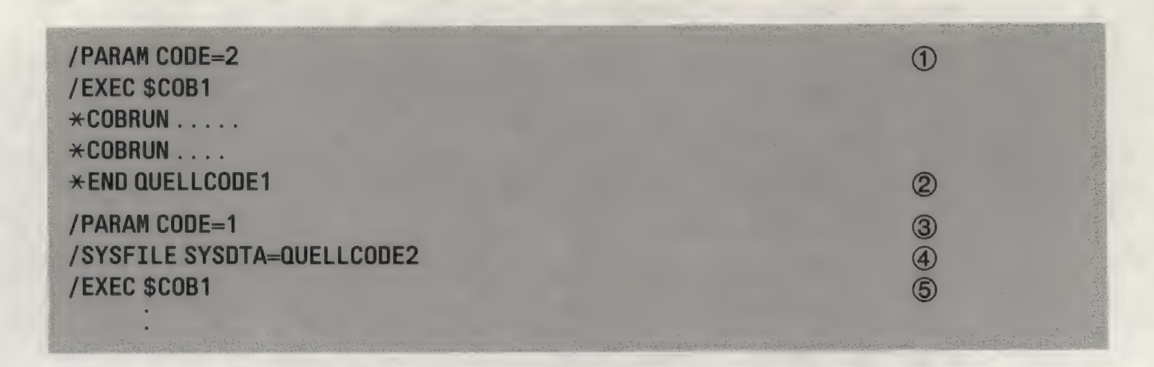
① Der Systemdatei SYSDTA wird die Datei QUELL.EINXEINS zugewiesen, in der sich das Quellprogramm befindet, das übersetzt werden soll.

② Der COB1-Übersetzer wird geladen und gestartet. Er verarbeitet die Daten, die von SYSDTA kommen.

③ Die Systemdatei SYSDTA wird für nachfolgende Aufgaben wieder der Datenstation zugewiesen.

Diese Eingabeform für ein COBOL-Quellprogramm wird am häufigsten verwendet.

Beispiel 14: Indirekte Eingabe aus einer Quellprogramm-Datei



```
/PARAM CODE=2                          ①
/EXEC $COB1
*COBRUN.....
*COBRUN....
*END QUELLCODE1                        ②
/PARAM CODE=1                          ③
/SYSFILE SYSDTA=QUELLCODE2             ④
/EXEC $COB1                            ⑤
     .
     .
```

① Der anschließend aufgerufene Übersetzer soll mit COBRUN-Operanden gesteuert werden.

② Die Eingabe von COBRUN-Operanden ist beendet und der Übersetzer soll den zu bearbeitenden Quellcode über SYSDTA aus der BS2000-Datei QUELLCODE1 lesen.

③ Nach Verarbeitung der Datei QUELLCODE1 hat der COB1-Übersetzer SYSDTA wieder auf PRIMARY zurückgesetzt. Das nächste BS2000-Kommando wird bearbeitet.

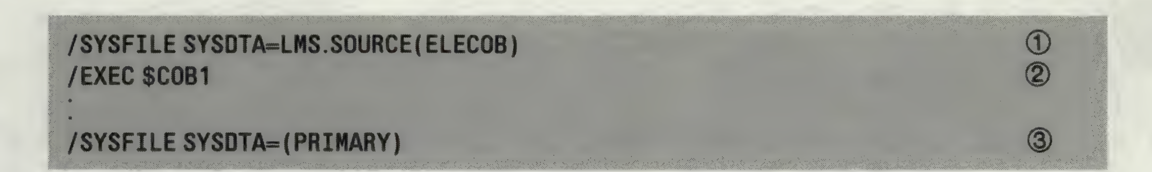
PARAM CODE=1 legt fest, daß der nächste Übersetzer keine Übersetzer-Steueranweisungen bearbeiten soll.

④ Die Systemdatei SYSDTA wird der Datei QUELLCODE2 zugewiesen.

⑤ Der COB1-Übersetzer wird aufgerufen und bearbeitet den Inhalt der Datei QUELLCODE2 mit Standardübersetzungs-Parametern.

Beispiel 14a: COBOL-Programm aus einer LMS-Quellprogramm-Bibliothek übersetzen

Unter dem Elementnamen ELECOB ist in der LMS-Quellprogramm-Bibliothek LMS.SOURCE das Programm mit dem PROGRAM-ID-Namen BEISP3 abgelegt. Zur Einleitung der Übersetzung sind folgende Kommandos bzw. Anweisungen nötig:



```
/SYSFILE SYSDTA=LMS.SOURCE(ELECOB)                    ①
/EXEC $COB1                                           ②
:
/SYSFILE SYSDTA=(PRIMARY)                             ③
```

Erläuterung:

① Die Systemdatei SYSDTA wird der Quellprogrammbibliothek LMS.SOURCE zugewiesen und darin dem Element mit Namen ELECOB.

② Aufruf des COB1-Übersetzers. Er greift direkt auf die zugewiesene Datei zu.

③ SYSDTA erhält wieder die Primärzuweisung.

Beispiel 14b: Indirekte Eingabe aus einer LMS-Programmbibliothek

Unter dem Elementnamen BIBELEM1 ist in der Programmbibliothek PLAMBIB1 das Quellprogramm mit dem PROGRAM-ID-Namen BEISP4 abgelegt. Zur Einleitung des Übersetzungslaufs sind folgende Kommandos bzw. Anweisungen nötig:

```
/FILE PLAMBIB1, LINK=SRCLIB                ①
/PARAM CODE=2                              ②
/EXEC $COB1
*COBRUN SRCELEM=BIBELEM1                   ③
*END                                       ④
```

Erläuterung:

① Die Programmbibliothek mit dem Namen PLAMBIB1 wird zugewiesen und mit dem Standardlinknamen SRCLIB verknüpft.

② Es sollen COBRUN-Steueranweisungen gelesen werden.

③ Das zu übersetzende COBOL-Programm steht unter dem Elementnamen BIBELEM1 in der mit FILE-Kommando zugewiesenen LMS-Programmbibliothek.

④ Die Eingabe von COBRUN-Anweisungen ist abgeschlossen. COB1 beginnt mit der Bearbeitung des Quellcodes.

2.2.4 Eingabe von Quellprogrammteilen

Quellprogrammteile (COPY-Elemente) können getrennt vom Quellprogramm, in dem sie Verwendung finden, in Bibliotheken gespeichert werden. Dies empfiehlt sich vor allem, wenn identische Teile in verschiedenen Quellprogrammen vorkommen. (Kapitel 1 erklärt, wie der Benutzer solche Daten in eine Bibliothek mit Hilfe des Dienstprogrammes LMS eingeben kann.)

Im Quellprogramm selbst steht stellvertretend für diese Programmteile eine COPY-Anweisung. COPY-Anweisungen dürfen an beliebiger Stelle im Quellprogramm stehen.

Vor dem Aufruf des Übersetzers muß die Bibliothek, in der sich die Quellprogrammteile befinden, mit einem FILE-Kommando zugewiesen und mit dem Dateikettungsnamen (LINK-Namen) verknüpft werden. Beim Erkennen der COPY-Anweisung im zu übersetzenden Quellprogramm fügt der Übersetzer dann aus der zugewiesenen Bibliothek das Element ein, dessen Name in der COPY-Anweisung angegeben wird. Der COPY-Elementtext wird dabei so übersetzt, als wäre er im Quellprogramm selbst geschrieben worden.

Quellcode kann in einer LMS-Bibliothek im Bibliotheksteil „Quellprogramm-Bibliothek" oder „Programmbibliothek" stehen. Im Teil „Programmbibliothek" muß das Quellcode-Element mit dem Typ S eingetragen sein.

Die COPY-Anweisung (siehe auch Manual „COB1 Beschreibung" [1]) im COBOL-Quellprogramm hat folgendes Format:

```
                 { OF }                                                     { wort-2        }
COPY textname [  {    } bibliotheksname] [SUPPRESS] [ REPLACING wort-1 BY  { literal-1     }
                 { IN }                                                     { bezeichner-1  }

                                                           { wort-4       }
                                             [wort-3 BY    { literal-2    } ] ... ] .
                                                           { bezeichner-2 }
```

textname (früher „elementname")
: 1 bis 30 Zeichen lang, bestehend aus Buchstaben, Ziffern und Bindestrich. Die ersten 8 Zeichen müssen eindeutig sein und den Konventionen des jeweiligen Bibliothekssystems entsprechen. Das erste Zeichen muß ein Buchstabe sein. Als letztes oder achtes Zeichen ist der Bindestrich unzulässig. Zwischenraum und Punkt dürfen nicht verwendet werden.

bibliotheksname
: 1 bis 30 Zeichen lang, bestehend aus Buchstaben und Ziffern. Die ersten 8 Zeichen müssen ein gültiger Linkname sein. Dieser Name muß auch im FILE-Kommando vor dem Aufruf des COB1-Übersetzers als LINK-Name angegeben werden:

  /FILE dateiname,LINK=bibliotheksname

  (wobei „dateiname" diejenige Datei bezeichnet, in welcher der zu kopierende Text steht).

  Stehen in einem Quellprogramm mehrere COPY-Anweisungen, so sind unterschiedliche Bibliotheksnamen zulässig. Entsprechend sind mehrere FILE-Kommandos abzusetzen.

  Fehlt die Angabe von bibliotheksname, so arbeitet COB1 mit Standard-Linknamen, für die vor Aufruf des COB1-Übersetzers die erforderlichen Bibliotheksdateien mit einem FILE-Kommando zugewiesen worden sein müssen.

  Steht das zu kopierende Bibliothekselement in einer bestimmten LMS-Quellprogramm- oder LMS-Programm-Bibliothek kann diese über den Link-Namen COBLIB zugewiesen werden. Es besteht auch die Möglichkeit, vor Beginn der Übersetzung über die Standard-Linknamen COBLIB, COBLIB1—COBLIB4 bis zu 5 LMS-Programm-Bibliotheken (PLAM-Bibliotheken) zuzuweisen. Diese durchsucht COB1 hierarchisch von COBLIB über COBLIB1 bis COBLIB4, bis das zu kopierende Element gefunden wurde.

SUPPRESS
: Dieser Zusatz bewirkt, daß die Protokollierung des COPY-Elements in der Quellprogrammliste unterdrückt wird. Die Angabe erübrigt sich, wenn eine COBRUN-Anweisung mit dem Operanden NOCOPY gegeben wurde.

REPLACING
: Dieser Zusatz bewirkt, daß im kopierten Bibliothekstext jedes im Original-Bibliothekstext enthaltene „wort-1" durch das ersetzt wird, was hinter BY angegeben ist. Der Originaltext in der Bibliothek bleibt unverändert.

wort-1/2/3/4 literal-1/2 bezeichner-1/2
: Jedes beliebige COBOL-Wort, jeder Datenname, Prozedurname, Bedingungsname, Merkname oder Dateiname, jede Zeichenfolge ist zulässig.

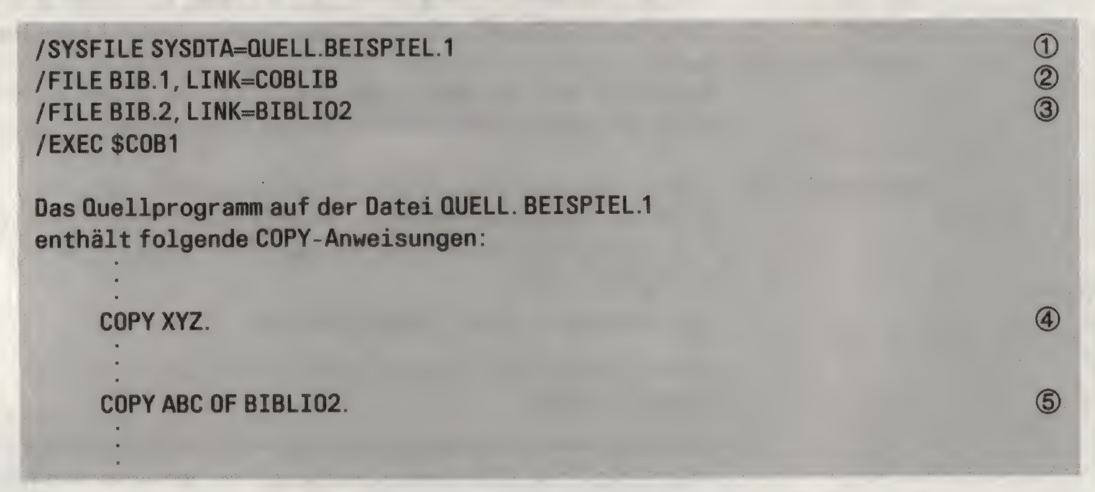
Regeln:

- Die Syntax des Textes wird vom Übersetzer im Rahmen des gesamten Quellprogramms geprüft.
- In der Quellprogrammliste ist der Bibliothekstext nach der zugehörigen COPY-Anweisung aufgelistet, gekennzeichnet durch

  ein „C“ vor der Nummer jeder Quellprogrammzeile.

Die Auflistung der COPY-Elemente in der Quellprogrammliste kann durch den Zusatz SUPPRESS in der COPY-Anweisung oder durch den Operanden NOCOPY in einer COBRUN-Anweisung unterdrückt werden.

Beispiel 15: Eingabe von Quellprogrammteilen



```
/SYSFILE SYSDTA=QUELL.BEISPIEL.1                ①
/FILE BIB.1, LINK=COBLIB                        ②
/FILE BIB.2, LINK=BIBLIO2                       ③
/EXEC $COB1

Das Quellprogramm auf der Datei QUELL.BEISPIEL.1
enthält folgende COPY-Anweisungen:
        .
        .
        .
    COPY XYZ.                                   ④
        .
        .
        .
    COPY ABC OF BIBLIO2.                        ⑤
        .
        .
        .
```

Erläuterung:

① SYSDTA wird der Eingabedatei QUELL.BEISPIEL.1 zugewiesen. Von dort erhält der Übersetzer ein Quellprogramm, in dem 2 Programmteile durch COPY-Anweisungen ersetzt sind.

② Das FILE-Kommando weist die Bibliothek BIB.1 zu und verknüpft sie mit dem Standard-Linknamen COBLIB.

③ Das zweite FILE-Kommando weist die Bibliothek BIB.2 zu und verknüpft sie mit dem Linknamen BIBLIO2 (=bibliotheksname in ⑤).

④ Der Text mit dem Namen XYZ wird aus der Bibliothek BIB.1 in das Quellprogramm kopiert. (Siehe ②.)

Handelt es sich bei BIB.1 um eine LMS-Programmbibliothek, wird das Element mit dem Namen XYZ, dem Elementtyp S und der alphanumerisch größten Version kopiert.
Wird der Text in der Bibliothek mit dem Standard-Linknamen COBLIB nicht gefunden, so werden auch die Bibliotheken mit den Standard-Linknamen COBLIB1 bis COBLIB4 danach durchsucht, sofern sie mit FILE-Kommandos zugewiesen wurden (COPY-Suchhierarchie).

⑤ Der Text mit dem Namen ABC wird aus der Bibliothek mit dem Dateinamen BIB.2 und dem Linknamen BIBLIO2 in das Quellprogramm kopiert. (Siehe ③.)

2.2.5 Eingabe von Steueranweisungen

Ablauf und **Ausgabe** des COB1-Übersetzers steuert der Benutzer durch

- PARAMETER-Kommando
- COBRUN-Anweisungen.

Das PARAMETER-Kommando gilt für einen ganzen Prozeß. Seine ausführliche Beschreibung steht im Abschnitt 2.4.2.

Dagegen gelten COBRUN-Anweisungen nur für die Übersetzung in der sie gegeben werden. Sie werden im folgenden beschrieben. (Siehe auch Abschnitt 2.4.3)

Format einer COBRUN-Anweisung:

COBRUN Operand 1 [,Operand 2]...

Zum Beispiel: COBRUN DIAGTEXT=GERMAN,ERRLST
COBRUN LOW#UP, TRUNCATE=NO

Damit der COB1-Übersetzer COBRUN-Anweisungen anfordert, muß der Benutzer vor dem Aufruf des COB1 das Kommando

/PARAM CODE=2

geben. Nach Aufruf des COB1-Übersetzers gibt man dann COBRUN-Anweisungen über die Systemdatei SYSDTA ein. Je nach der Zuweisung von SYSDTA können diese Anweisungen entweder direkt am Datensichtgerät oder aus einer Datei eingegeben werden.

Zu beachten ist:

- Die COBRUN-Anweisungen müssen mit der END-Anweisung abgeschlossen werden.
- folgende Reihenfolge ist einzuhalten:
  COBRUN-Anweisungen
  END-Anweisung
  Quellprogramm
- Kommen COBRUN-Anweisungen und Quelldaten von verschiedenen Eingabequellen, so muß SYSDTA nach den COBRUN-Anweisungen umgewiesen werden, indem man in der END-Anweisung den Namen der Datei oder der Bibliothek und des Elements angibt, aus der weitere Quelldaten gelesen werden sollen. Am Ende der Übersetzung setzt COB1 SYSDTA wieder auf (SYSCMD) zurück.

Format der END-Anweisung:

```
END [{dateiname                }]
    [{biblname(elementname)    }]
```

Die folgende Übersicht beschreibt in Kurzform die Aufgaben der verschiedenen COBRUN-Operanden. Ihre ausführliche Beschreibung ist in Abschnitt 2.4.3 zu finden.

COBRUN-Operanden steuern den **Ablauf der Übersetzung:**

QUOTE1	veranlaßt, daß ein Apostroph (') im Quellprogramm als Anführungszeichen (") interpretiert wird (siehe /PARAM-Schlüsselwort CODE=3).
SEMCHK	legt fest, daß das Quellprogramm nur syntaktisch und semantisch überprüft und kein Bindemodul erzeugt wird.
SYNCHK	bewirkt den Abbruch des Übersetzungsvorgangs, nachdem das Quellprogramm ausschließlich auf syntaktische Fehler geprüft worden ist.
SRCELEM= elementname	nennt den Elementnamen, unter dem in einer LMS-Programmbibliothek das zu übersetzende COBOL-Quellprogramm abgespeichert ist. Diese Bibliothek muß mit FILE-Kommando über den LINK-Namen SRCLIB zugewiesen sein.

COBRUN-Operanden beeinflussen die **Form des übersetzten Programms** (Bindemodul):

ACTKEY = { TTTR / TTTT }	legt den Wertebereich für den ACTUAL KEY fest. (Betrifft nur die direkte Dateiorganisation)
LINK	bewirkt, daß der LINK-Name für Dateien aus dem Herstellerwort der ASSIGN-Klausel (siehe diese in [1]) statt aus dem Dateinamen der SELECT-Klausel entnommen wird.
LOW#UP	bewirkt bei Ausführung einer ACCEPT-Anweisung, daß eingegebene Kleinbuchstaben in Großbuchstaben umgesetzt werden.
NESTPF	Der Übersetzer generiert Befehlsfolgen für PERFORM-Anweisungen, die auf einen gemeinsamen EXIT führen. Achtung: Der Operand erzeugt eine umfangreichere Programmsteuerung. Dies kann zu Einbußen bei Laufzeit und zu längeren Objekten führen. Beispiel: : PERFORM A THRU C. ① : A. : PERFORM B THRU C. ② : B. C. EXIT. ← Dieser EXIT ist für ① und ② gemeinsam.
PARAM8	hat zur Folge, daß nach einer MOVE-Anweisung auf numerische Felder aus Leerstellen die Ziffer Null erzeugt wird.
RANGECHECK= { CONTINUE / TERM / NO }	Zur Programmlaufzeit kann die Einhaltung von Tabellengrenzen beim Ansprechen von Tabellenelementen überprüft werden. CONTINUE: Nur Meldung des eigentlich unzulässigen Zugriffs. TERM: Meldung der versuchten Indexgrenzenverletzung und Programmabbruch. NO: Keine Überprüfung (Standard).
SSEQ#GEN	die Meldungen 90xx und 91xx werden ergänzt mit der von COB1 vergebenen Quellprogramm-Zeilennummer der Anweisung, bei deren Ausführung die Meldung ausgegeben wurde.
SYMTEST = { ALL / MAP / NO }	legt die Art der Informationen fest, die der Übersetzer für die Dialogtesthilfe AID bereitstellt. Diese Informationen werden dem Bindemodul übergeben. Sie lassen sich in zwei Teile gliedern, — eine „List for Symbolic Debugging" (LSD), in der die innerhalb des Moduls definierten symbolischen Namen verzeichnet sind und — ein „External Symbol Dictionary" (ESD), welches die Externbezüge des Moduls registriert. (Zum Format dieser Informationen siehe [23].) ALL: Dieser Wert muß gesetzt werden, wenn das Programm mit der Dialogtesthilfe AID symbolisch überwacht werden soll. Der Übersetzer erzeugt dann sowohl LSD- als auch ESD-Informationen, so daß beim Testen mit AID symbolische Namen aus dem Quellprogramm (wie in [25] beschrieben) verwendet werden können. Hinweise: 1. Durch COBRUN SYMTEST=ALL wird ein evtl. bestehendes PARAM SYMDIC=YES außer Kraft gesetzt, d. h. ISD-Informationen werden nicht erzeugt. 2. Bei segmentierten Programmen ist die Erzeugung von LSD-Informationen (und damit symbolisches Testen mit AID) nur dann möglich, wenn der Bindemodul mit COBRUN MODULE = bibliotheksname in eine PLAM-Bibliothek ausgegeben wird.

	MAP: Der Übersetzer erzeugt lediglich ESD-Testhilfeinformationen vom Typ Übersetzungseinheit. Dabei wird dem Objektmodul (bei segmentierten Programmen: allen Moduln) ein symbolischer Name zugeordnet, der aus den ersten 8 Zeichen des Namens im PROGRAM-ID-Paragraphen besteht. Beim Testen mit AID kann dieser Name zur Qualifikation des Quellprogrammes verwendet werden. Eine darüber hinausgehende Programmüberwachung auf symbolischer Ebene mit AID ist nicht möglich. NO: Dieser Wert ist nur zu setzen, wenn das Binder-/Ladersystem Objektmoduln mit ESD-Einträgen vom Typ Übersetzungseinheit nicht verarbeiten kann. Der Übersetzer gibt dann weder LSD- noch ESD-Testhilfeinformationen an den Bindemodul weiter. Symbolisches Testen mit AID ist nicht möglich. Standardwert ist SYMTEST = MAP.
TCBENTRY=name	Name der Verbindungstabelle zu UTM (vgl. [10], [14]), die COB1 erzeugt. Diese Tabelle enthält Zeiger zu internen, vom Übersetzer erzeugten Arbeitsbereichen, die bei Wiederdurchlauf eines UTM-Teilprogramms von UTM erneut initialisiert werden müssen. „name" kennzeichnet den Anfang dieser Zeiger-Tabellen.
TRUNCATE = { YES / ALL / NO }	legt fest, wie sich die in der PICTURE-Klausel angegebene Dezimalstellenanzahl bei Datenfeldern auswirkt, für die USAGE IS COMPUTATIONAL vereinbart wurde: YES: Bei der Übertragung eines numerischen Literals in ein binäres Datenfeld wird nur die Anzahl Dezimalstellen berücksichtigt, die in der PICTURE-Klausel angegeben wurde. Überschüssige Dezimalziffern werden ggf. abgeschnitten. Beispiel: 77 EMPF PIC S999 USAGE IS COMPUTATIONAL. MOVE 1234 TO EMPF. Inhalt von EMPF: 234 ALL: Über die bei TRUNCATE = YES angegebenen Fälle hinaus wird auch bei allen MOVE-Anweisungen in binäre Felder nur die Anzahl Dezimalstellen berücksichtigt, die in der PICTURE-Klausel vereinbart wurde. Überschüssige Dezimalziffern werden ggf. abgeschnitten. NO: Bei der Übertragung eines numerischen Literals in ein binäres Datenfeld wird die Anzahl Binärstellen berücksichtigt, die für das Empfangsfeld über die PICTURE-Klausel reserviert wurde. Es werden nur dann Binärstellen des Literals abgeschnitten, wenn ihre Anzahl die tatsächliche Länge des Empfangsfeldes übersteigt. Beispiele: 1. 77 EMPF PIC S999 USAGE IS COMPUTATIONAL. MOVE 1234 TO EMPF. Inhalt von EMPF: 1234 2. 77 EMPF PIC S999 USAGE IS COMPUTATIONAL VALUE IS 65536. Inhalt von EMPF: 0 Standardwert ist TRUNCATE = YES.

COBRUN-Operanden steuern die **Listen- oder Modulausgabe** des COB1-Übersetzers:

ERRLST	erlaubt die Unterdrückung von Objektprogramm-, Adreß- und Querverweislisten bei Fehlern mit SEVERITY CODE größer oder gleich 2.
LINEnn	legt für die Listenausgabe die maximale Zeilenanzahl nn pro Seite fest (Standard: nn = 64).
MAPALL	veranlaßt die Ausgabe einer doppelten Adreßliste, die einmal aufsteigend nach Quellprogrammfolgenummern (Standard) und einmal alphabetisch nach Datennamen sortiert ist.
MAPSRT	bewirkt die Ausgabe einer Adreßliste, die nur nach Datennamen sortiert ist.
MODULE= bibliotheks-name	steuert die Ausgabe eines Objektmoduls in eine LMS-Programmbibliothek. bibliotheksname = Dateiname einer PLAM-Bibliothek. (Dieser Operand ist wirkungslos, falls gleichzeitig /PARAM CARD=NO, /PARAM DISC=NO oder COBRUN SYNCHK gilt.)

NOCOPY	unterdrückt die Protokollierung aller COPY-Elemente. Sollen nur einzelne Elemente nicht protokolliert werden, so ist nicht dieser Operand, sondern in der betreffenden COPY-Anweisung der Zusatz SUPPRESS anzugeben.
NODDLIST	Unterdrückt die Protokollierung der SUB-SCHEMA SECTION des Quellprogramms in der Quellprogrammliste.
WRLST	bewirkt die Ausgabe der Listen in die Systemdatei SYSLST.

COBRUN-Operanden steuern die vom COB1-Übersetzer erzeugten **Meldungen:**

DIAGTEXT = {ENGLISH / GERMAN}	erlaubt die Wahl zwischen englischen und deutschen Fehlermeldungstexten.
ERDICT	veranlaßt die Ausgabe einer Liste sämtlicher Fehlermeldungen des COB1-Übersetzers.
ERRPRn	unterdrückt Fehlermeldungen für Fehler mit einem SEVERITY CODE kleiner als n. Für n können folgende Werte angegeben werden: n=I Alle Fehler der Fehlerklassen I, 0, 1, 2, 3 werden protokolliert. n=0 Alle Fehler der Fehlerklassen 0, 1, 2, 3 werden protokolliert. n=1 Alle Fehler der Fehlerklassen 1, 2, 3 werden protokolliert. n=2 Alle Fehler der Fehlerklassen 2, 3 werden protokolliert. n=3 Alle Fehler der Fehlerklassen 3 werden protokolliert. Fehlt dieser Operand, wird n = I angenommen. Hinweis: Unterdrückte Fehlermeldungen werden mitgezählt. In der Fehlerliste wird die Meldung >PRINTING SUPPRESSED< ausgegeben.
SEQERR	veranlaßt, daß eine Fehlermeldung in die Fehlerliste ausgegeben wird, sobald Quellprogrammsätze nicht in aufsteigender Reihenfolge vorgefunden werden.

Beispiel 17: Eingabe von COBRUN-Anweisungen

```
/PARAM CODE=2                                                        ①
/SYSFILE SYSDTA=(SYSCMD)
/EXEC $COB1
% P500 PROGRAM COB1, VERSION 22B OF 85-03-08 LOADED.
*COBRUN DIAGTEXT=GERMAN                                              ②
9099 COBRUN DIAGTEXT=GERMAN
*END QUELL. EINXEINS                                                 ③
9099 END QUELL. EINXEINS
9017 BEGINN DER UEBERSETZUNG, VERSION V02. 2B
9097 DIE UEBERSETZUNG WURDE OHNE FEHLER BEENDET
9004 DIE UEBERSETZUNG VON EINXEINS BENOETIGTE 00, 55 CPU SEKUNDEN
/PARAM CODE=1                                                        ④
```

① Die Vereinbarung CODE=2 legt fest, daß der COB1-Übersetzer COBRUN-Anweisungen verarbeiten soll.

② Aus der Systemdatei SYSDTA (hier: Datenstation) erwartet der COB1-Übersetzer die erste COBRUN-Anweisung.

③ Die Eingabe von COBRUN-Anweisungen wird abgeschlossen und der Systemdatei SYSDTA die Quellprogramm-Datei QUELL.EINXEINS zugewiesen, deren Übersetzung damit eingeleitet wird.

④ Der CODE-Wert wird auf Standardwert zurückgesetzt, so daß bei evtl. folgenden Übersetzungen keine COBRUN-Anweisungen mehr erwartet werden.

2.3 Ausgabe

2.3.1 Übersicht über Ausgabemöglichkeiten

Der COB1-Übersetzer kann während seines Ablaufs folgende Ausgaben erzeugen:

- **Bindemoduln**, d.h. die Übersetzung des Quellprogrammes in Maschinensprache (Objektprogramm). Wahlweise wird dem Bindemodul ein Internadreßbuch (ISD, Internal Symbol Dictionary) bzw. eine „List for Symbolic Debugging" (LSD) angefügt. Dadurch kann der Benutzer zur Programmablaufzeit die Dialogtesthilfe IDA (Interactive Debugging Aid) bzw. AID (Advanced Interactive Debugger, siehe 5.3 und [25]) mit den symbolischen Adressen des Quellprogrammes einsetzen.
- **Listen**, die das eingegebene Quellprogramm und das erzeugte Objektprogramm protokollieren, sowie Listen der im Programm verwendeten Adressen und Querverweise. Außerdem wird eine Liste aller während der Übersetzung im Quellprogramm festgestellten Fehler erzeugt:

Steueranweisungsliste	(COBRUN/PARAM LISTING)
Quellprogrammliste	(SOURCE LISTING)
Objektprogrammliste	(OBJECT PROGRAM LISTING)
Adreßliste Querverweisliste	(LOCATOR/MAP LISTING)
Fehlermeldeliste	(DIAGNOSTIC LISTING)

- **Meldungen** über den Ablauf der Übersetzung; sie gehen nach SYSOUT. Anhang 1 enthält die Liste der vom Übersetzer ausgegebenen „90/91er-Meldungen".

Bindemoduln werden standardmäßig in die **temporäre Bindemoduldatei** * des laufenden Prozesses gebracht. Ist COBRUN MODULE=bibliotheksname angegeben, erfolgt die Ausgabe der Bindemoduln in eine Programmbibliothek (LMS [21]).
Die Sicherstellung der Bindemoduln wird in Kapitel 3 behandelt.

Bei Fehlermeldungen und Listen kann der Benutzer zwischen der Ausgabe in temporäre und katalogisierte Dateien wählen. Die **temporären Dateien** gibt das System automatisch auf den Drucker aus, wobei der Benutzer (mit dem COBRUN-Operanden WRLST) noch entscheiden kann, ob die Ausgabe bereits während des Übersetzerlaufs oder erst bei Prozeßende (LOGOFF-Bearbeitung) eingeleitet werden soll. In beiden Fällen werden die temporären Dateien nach erfolgter Ausgabe automatisch gelöscht. Im Gegensatz dazu kann der Benutzer die Informationen in den **katalogisierten Dateien** zu einem beliebigen Zeitpunkt mit Hilfe des PRINT-Kommandos [2] ausgeben lassen.

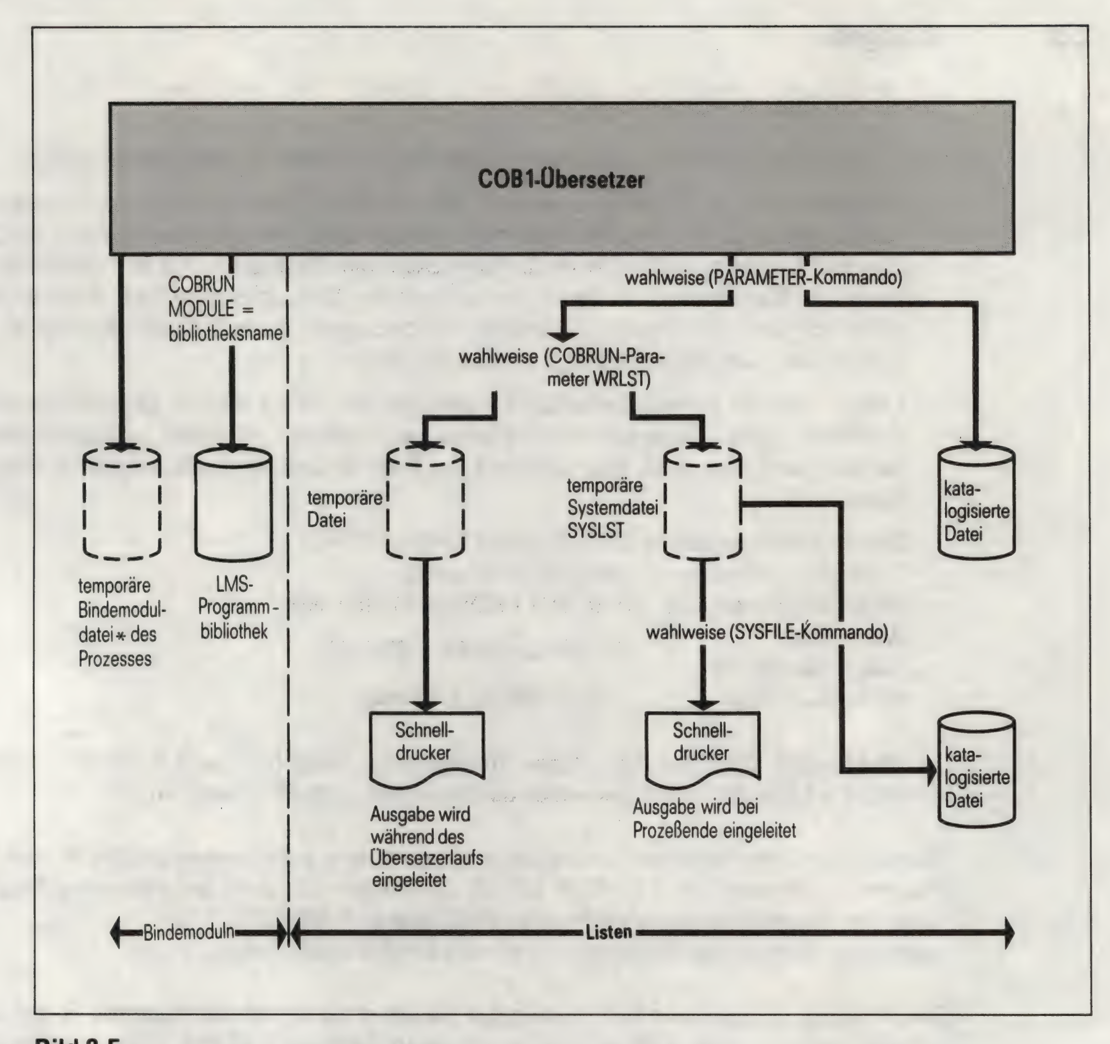


Bild 2-5

Ausgabe von Bindemoduln und Listen

2.3.2 Betriebsmittelzuweisung für die Ausgabe

Zu den Betriebsmitteln für die Ausgaben des COB1-Übersetzers zählen katalogisierte Dateien und temporäre Dateien. Bei Fehlermeldungen und Listen kann der Benutzer den Zeitpunkt der Ausgabe bestimmen und die Betriebsmittel auswählen. Der COB1-Übersetzer macht standardmäßig Vorgaben, die der Benutzer abändern kann, und zwar mit folgenden BS2000-Kommandos und COB1-Steueranweisungen:

- PARAMETER-Kommando,
- FILE-Kommando,
- COBRUN-Anweisung mit Operand MODULE=bibliotheksname
- COBRUN-Anweisung mit Operand WRLST,
- SYSFILE-Kommando mit Operand SYSLST.

Mit dem **PARAMETER-Kommando** kann der Benutzer entscheiden, in welchen Dateityp bzw. ob überhaupt ausgegeben werden soll. Das Format dieses Kommandos lautet:

/PARAM[ETER] schlüsselwort1=wert1[,schlüsselwort2=wert2]...

Abschnitt 2.4.2 enthält die ausführliche Beschreibung sämtlicher Schlüsselwörter des PARAM-Kommandos, die folgende Übersicht zeigt nur diejenigen Schlüsselwörter, die Betriebsmittel für die Ausgabe zuweisen. Standardwerte sind unterstrichen; sie gelten, wenn der Benutzer keine Angaben macht.

Schlüsselwörter des PARAMETER-Kommandos für die Zuweisung der Betriebsmittel:

Name der Ausgabe	Ausgabe in temporäre Datei	Ausgabe in katalogisierte Datei	Unterdrückung der Ausgabe	Unterdrückung bei Fehlern mit SEVERITY CODE ≥ 2
Fehler- meldungen (DIAGNOSTIC LISTING/FILE)	DIAG = YES	ERRFIL = YES Dateiname: ERRFIL.COB1. progname	DIAG = NO und ERRFIL = NO	–
Quellpro- grammliste (SOURCE LISTING)	LIST = YES	SAVLST = SOURCE oder SAVLST = ALL Dateiname: SRCLST.COB1. progname	LIST = NO und SAVLST = NO	– –
Objektpro- grammliste (OBJECT PROGRAM LISTING)	OBJLST = YES	SAVLST = OBJECT oder SAVLST = ALL Dateiname: OBJLST.COB1. progname	OBJLST = NO und SAVLST = NO	ERRLST = NO*)
Adreßliste (LOCATOR/ MAP LISTING)	MAP = YES	SAVLST = LOCMAP oder SAVLST = ALL Dateiname: LOCLST.COB1. progname	MAP = NO und SAVLST = NO	ERRLST = NO*)
Querver- weisliste (LOCATOR/ MAP LISTING)	XREF = YES	SAVLST = LOCMAP oder SAVLST = ALL mit XREF = YES Dateiname: LOCLST.COB1. progname	XREF = NO und SAVLST = NO	ERRLST = NO*)

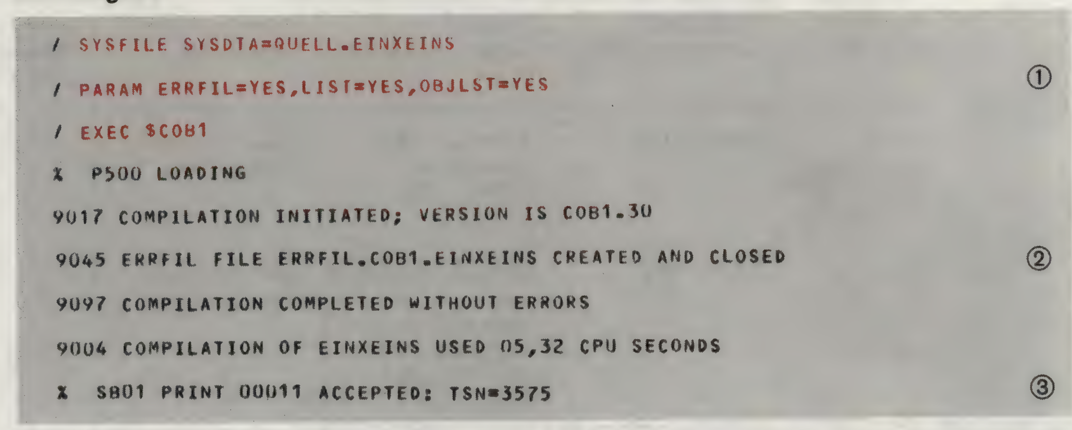
*) Gleiche Wirkung hat der COBRUN-Operand ERRLST.

Standardmäßig vorgegeben sind also folgende Funktionen:

1. die Ausgabe des Objektprogramms in die temporäre Bindemoduldatei „*" des Prozesses,
2. die Ausgabe der Fehlermeldungsliste auf den Drucker,
3. die Ausgabe der Adreßliste auf den Drucker.

Sie sind in einem Prozeß so lange gültig, bis der Benutzer sie durch ein oder mehrere PARAMETER-Kommandos abändert. Ein PARAM-Kommando beeinflußt dabei nur die Schlüsselwörter, die in ihm explizit angegeben werden, d.h. es setzt die übrigen Schlüsselwörter nicht auf ihre Standardwerte zurück. Dies kann der Benutzer in einer Prozedurdatei durch das STEP-Kommando [2] erreichen.

Beispiel 18: Verwendung des PARAMETER-Kommandos bei der Steuerung der Übersetzer-Ausgabe



```
/ SYSFILE SYSDTA=QUELL.EINXEINS
/ PARAM ERRFIL=YES,LIST=YES,OBJLST=YES                         ①
/ EXEC $COB1
% P500 LOADING
9017 COMPILATION INITIATED; VERSION IS COB1.30
9045 ERRFIL FILE ERRFIL.COB1.EINXEINS CREATED AND CLOSED       ②
9097 COMPILATION COMPLETED WITHOUT ERRORS
9004 COMPILATION OF EINXEINS USED 05,32 CPU SECONDS
% S801 PRINT 00011 ACCEPTED: TSN=3575                          ③
```

① Das PARAMETER-Kommando bestimmt, daß eine Fehlerdatei (ERRFIL) erzeugt wird. Außerdem erzeugt der Übersetzer eine Quellprogramm- und eine Objektprogrammliste.

② Der COB1-Übersetzer meldet die Erzeugung der Fehlerdatei ERRFIL.COB1.EINXEINS (EINXEINS ist der Programmname).

③ Das System meldet, daß ein von der Übersetzung unabhängiger Prozeß mit der Prozeßfolgenummer (TSN) 3575 zum Ausdrucken der Listen initialisiert wurde.

Dateinamen und Dateieigenschaften für die katalogisierten Ausgabedateien des COB1-Übersetzers sind standardmäßig vorgegeben. Der Benutzer kann aber mit Hilfe des **FILE-Kommandos** diese Vorgaben abändern, zum Beispiel Dateinamen eigener Wahl vergeben. Dazu muß er vor dem Aufruf des Übersetzers die gewünschten Eigenschaften in einem FILE-Kommando mit den entsprechenden Dateikettungsnamen (LINK-Namen) verknüpfen, die der COB1-Übersetzer verwendet:

/FILE dateiname,LINK=dateikettungsname,...

Name der Ausgabedatei	Dateikettungsname des COB1-Übersetzers
Datei mit Quellprogrammliste	SRCLINK
Datei mit Objektprogrammliste	OBJLINK
Datei mit Adreßliste	LOCLINK
Fehlerdatei	ERRLINK

Beispiel 19: Verwendung eines Dateikettungsnamens

```
/ PARAM ERRFIL=YES
/ FILE FEHLER,LINK=ERRLINK                                      ①
/ SYSFILE SYSDTA=QUELL.EINXEINS.FEHLER
/ EXEC $RZ4.COB1.130
% P500 LOADING
9017 COMPILATION INITIATED; VERSION IS COB1.30
9045 ERRFIL FILE FEHLER CREATED AND CLOSED                      ②
9001 EINXEINS  TOTAL FLAGS:00004 /S0= 0000 /S1= 0003 /S2= 0001  ③
9004 COMPILATION OF EINXEINS USED 04,49 CPU SECONDS
% S801 PRINT 00009 ACCEPTED: TSN=3658
/RELEASE ERRLINK                                                ④
```

① Die vom Übersetzer zu erzeugende Fehlerdatei bekommt den Namen FEHLER anstelle des Standardnamens ERRFIL.COB1.programmname.

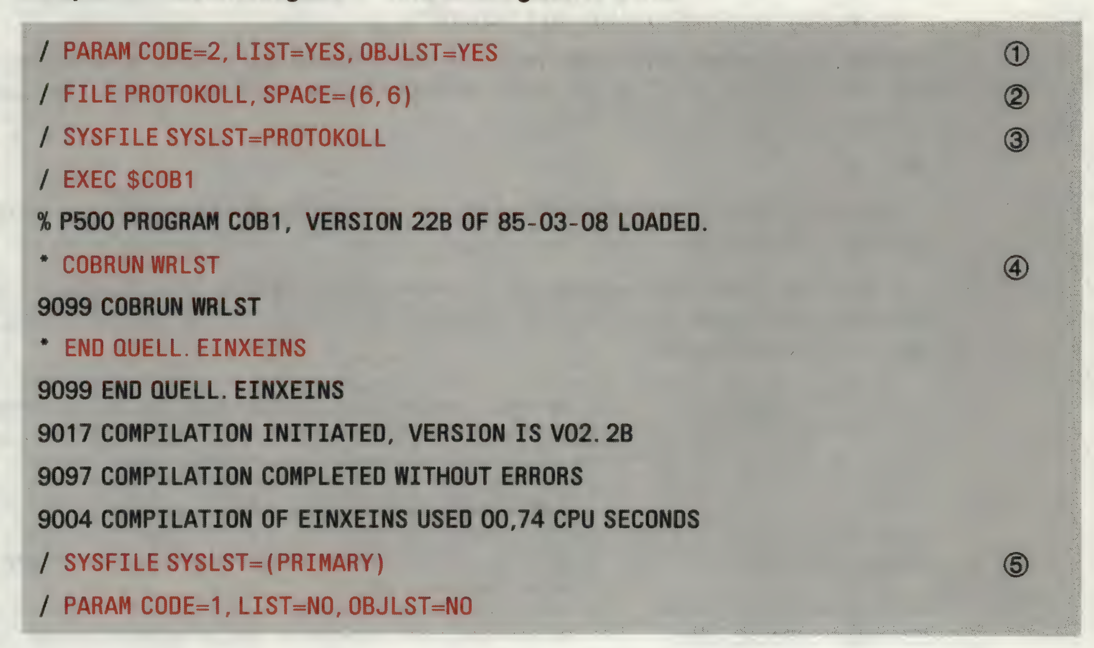
② Die Fehlerinformation wurde in die Datei FEHLER ausgegeben.

③ Der Übersetzer fand vier Fehler: drei in der Fehlerklasse 1 und einen in der Fehlerklasse 2.

④ Die Datei FEHLER soll beim nächsten Übersetzungslauf nicht mehr verwendet werden; ihre Verkettung mit dem Dateikettungsnamen ERRLINK wird deshalb hier gelöscht.

Durch eine COBRUN-Anweisung mit dem Operanden WRLST kann der Benutzer erreichen, daß die Listen in die Systemdatei SYSLST ausgegeben werden: Sämtliche Listen, deren Ausgabe standardmäßig bereits während des Übersetzungslaufes eingeleitet wird, werden also erst nach Prozeßende ausgespult. Das hat den Vorteil, daß der Benutzer noch während des Prozeßlaufs mit dem SYSFILE-Kommando [2] die Ausgabe auf SYSLST steuern kann.

Beispiel 20: Listenausgabe in eine katalogisierte Datei



```
/ PARAM CODE=2, LIST=YES, OBJLST=YES                        ①
/ FILE PROTOKOLL, SPACE=(6, 6)                              ②
/ SYSFILE SYSLST=PROTOKOLL                                  ③
/ EXEC $COB1
% P500 PROGRAM COB1,  VERSION 22B OF 85-03-08 LOADED.
* COBRUN WRLST                                              ④
9099 COBRUN WRLST
* END QUELL. EINXEINS
9099 END QUELL. EINXEINS
9017 COMPILATION INITIATED,  VERSION IS V02. 2B
9097 COMPILATION COMPLETED WITHOUT ERRORS
9004 COMPILATION OF EINXEINS USED 00,74 CPU SECONDS
/ SYSFILE SYSLST=(PRIMARY)                                  ⑤
/ PARAM CODE=1, LIST=NO, OBJLST=NO
```

① Für die nachfolgenden Übersetzungen wird vereinbart, daß der COB1-Übersetzer COBRUN-Anweisungen aus der Systemdatei SYSDTA einliest (CODE=2) und eine Objekt- und eine Quellprogrammliste ausgibt (OBJLST=YES, LIST=YES).

② Für die Datei PROTOKOLL weist das FILE-Kommando Speicherplatz zu.

③ Das SYSFILE-Kommando erklärt die Datei PROTOKOLL zur System-Ausgabedatei SYSLST.

④ Dem COB1-Übersetzer wird mitgeteilt, daß er die Listen nach SYSLST ausgeben soll. Wegen der zuvor getroffenen Vereinbarung ist SYSLST hier die Datei PROTOKOLL.

⑤ Schließen der Datei PROTOKOLL.
Diese katalogisierte Datei kann nun z. B. mit dem Kommando
/PRINT PROTOKOLL, SPACE=E
ausgedruckt werden.
Die Systemdatei SYSLST weist nun auf die temporäre (EAM-)Ausgabedatei des Prozesses, die bei Prozeßende automatisch ausgegeben wird.

2.3.3 Ausgabe von Bindemoduln

Der COB1-Übersetzer überträgt die eingegebenen Quelldaten in Maschinensprache und erzeugt auf diese Weise ein sogenanntes Objektprogramm, das aus einem oder mehreren Bindemoduln besteht. Der Benutzer kann veranlassen, daß dem Objektprogramm ein Internadreßbuch (ISD, **I**nternal **S**ymbolic **D**ictionary) oder ein symbolisches Adreßbuch (LSD, **L**ist for **S**ymbolic **D**ebugging) zugeordnet wird, das die symbolischen Adressen des Quellprogramms speichert.

Der COB1-Übersetzer gibt Bindemoduln standardmäßig in die temporäre EAM-Bindemoduldatei „*" des laufenden Prozesses aus, die auch von anderen Übersetzern (z. B. ASSEMB, FOR1) für diesen Zweck verwendet wird. Die Bindemoduln werden dort additiv, d. h. ohne Bezug zueinander, abgespeichert.

Die EAM-Bindemoduldatei „*" gehört zu dem Prozeß, in dem die Übersetzung stattfindet. Sie wird beim ersten Übersetzungslauf für diesen Prozeß angelegt und bei Prozeßende (LOGOFF-Bearbeitung) automatisch gelöscht. Soll das Ergebnis der Übersetzung also weiter verwendet werden, so ist der Benutzer dafür verantwortlich, daß der Inhalt der *-Datei sichergestellt bzw. weiterverarbeitet wird. Die verschiedenen Möglichkeiten, die ihm dabei offen stehen, sind in Kapitel 3 und 4 aufgeführt.

Werden die übersetzten Bindemoduln in der *-Datei nicht mehr benötigt, zum Beispiel, weil sie fehlerhaft sind, so empfiehlt es sich, die *-Datei spätestens vor dem nächsten Übersetzungslauf mit dem Kommando

/ERASE *

zu löschen. Dadurch wird vermieden, daß die fehlerhaften Bindemoduln den weiteren Verarbeitungsablauf stören.

Der Benutzer kann die Ausgabe von Objektmoduln und Internadreßbuch mit bestimmten Schlüsselwörtern des PARAMETER-Kommandos und mit Operanden der COBRUN-Steueranweisung beeinflussen:

COBRUN MODULE= bibliotheksname	bewirkt Ausgabe eines Objektprogramms in eine Programmbibliothek (LMS).
COBRUN SEMCHK	legt fest, daß das Quellprogramm nur syntaktisch und semantisch überprüft und kein Bindemodul erzeugt wird.
COBRUN SYNCHK	legt fest, daß das Quellprogramm nur syntaktisch überprüft und kein Bindemodul erzeugt wird.
/PARAM SYMDIC=YES, DEBUG=YES	bewirkt die Erzeugung des Internadreßbuchs, die Symbole der COBOL-Verben werden mit Hilfe der Folgenummern des Quellprogramms gebildet.
/PARAM SYMDIC=YES, DEBUG=NO	bewirkt die Erzeugung des Internadreßbuchs, die Symbole der COBOL-Verben werden mit Hilfe der vom Übersetzer erzeugten Nummern gebildet.
COBRUN ACTKEY, LINK, LOW#UP, NESTPF, PARAM8, RANGECHECK, TCBENTRY, SYMTEST	betreffen die Eigenschaften der von COB1 erzeugten Objektmoduln.

Eine ausführliche Beschreibung ist in den Abschnitten 2.4.2 und 2.4.3 zu finden.

Die Struktur eines Objektprogramms geht aus der vom COB1-Übersetzer ausgegebenen Objektprogrammliste hervor und wird im Anhang 3 dargestellt.

2.3.4 Ausgabe von Listen

Folgende Arten von Listen können vom COB1-Übersetzer erzeugt werden:

- Steueranweisungsliste
- Quellprogrammliste (mit Bibliotheksliste)
- Objektprogrammliste
- Adreßliste/Querverweisliste
- Fehlermeldungsliste.

Durch Angaben im PARAMETER-Kommando kann die Ausgabe einzelner Listen angefordert oder unterdrückt werden.
Außerdem kann der Benutzer durch Angabe von COBRUN-Anweisungen das Listenbild beeinflussen.

In diesem Abschnitt werden die möglichen Listen detailliert anhand eines Programmbeispiels beschrieben.

Auf jeder Seite der vom COB1-Übersetzer erzeugten Listen erscheint eine Überschriftzeile, die folgende Informationen enthält:

Ⓐ Name und Versionsnummer des Übersetzers

Ⓑ Kennzeichnung des ANS-Standards[1])

Ⓒ PROGRAM-ID-Name[2])

Ⓓ Listenart

Ⓔ Uhrzeit der Übersetzung

Ⓕ Datum der Übersetzung

Ⓖ Seitennummer

Nach jeder Überschriftzeile werden von COB1 zwei Leerzeilen erzeugt.

Mit dem PARAMETER-Kommando oder der COBRUN-Steueranweisung steuert der Benutzer die Ausgabe von Listen. Die folgende Übersicht zeigt die verschiedenen Möglichkeiten. Eine ausführliche Beschreibung der aufgeführten Schlüsselwörter des PARAM-Kommandos bzw. der COBRUN-Operanden befindet sich in den Abschnitten 2.4.2 und 2.4.3.

---

[1]) COBOL74 entspricht ANS74

[2]) erscheint nicht auf der zweiten Seite der Liste, da dort der PROGRAM-ID-Paragraph selbst steht

Tabelle: Ausgabe von Listen

Listentitel	Beschreibung	Schlüsselwörter des **PARAM-Kommandos** bewirken Ausgabe		Operanden der **COBRUN-Anweisung** steuern Ausgabe*)
		auf Drucker	in Datei	
COBRUN/PARAM LISTING	Steueranweisungsliste 2.3.4.1			
SOURCE LISTING	Quellprogrammliste Bibliotheksliste 2.3.4.2	LIST=YES	SAVLST= SOURCE oder SAVLST= ALL	NOCOPY unterdrückt die Protokollierung der COPY-Elemente, siehe auch SUPPRESS-Zusatz in COPY-Anweisung NODDLIST unterdrückt die Protokollierung der SUB-SCHEMA SECTION des Quellprogramms
OBJECT PROGRAMM LISTING	Externadreßbuch Tabellen- und Speicherbereiche Objektprogrammliste Internadreßbuch 2.3.4.3	OBJLST= YES	SAVLST= OBJECT oder SAVLST= ALL	ERRLST unterdrückt die Ausgabe bei Fehlern mit SEVERITY CODE ≥ 2
LOCATOR/MAP LISTING	Adreßliste der DATA DIVISION und PROCEDURE DIVISION 2.3.4.4	MAP=YES	SAVLST= LOCMAP oder SAVLST= ALL	MAPSRT sortiert nach Namen MAPALL sortiert nach Namen und Folgenummer ERRLST unterdrückt die Ausgabe bei Fehlern mit SEVERITY CODE ≥ 2
	Querverweisliste 2.3.4.4	XREF=YES	SAVLST= LOCMAP oder SAVLST= ALL mit XREF=YES	
DIAGNOSTIC LISTING	Fehlermeldungsliste 2.3.4.5	DIAG=YES	ERRFIL= YES	ERRPRn unterdrückt die Ausgabe von Fehlermeldungen für Fehler mit SEVERITY CODE $< n$

*) WRLST und LINEnn gelten für alle Listen.

2.3.4.1 Steueranweisungsliste

Hier protokolliert der COB1-Übersetzer

ⓐ die Umgebung des Übersetzungsprozesses,

ⓑ die ausgewählten COBRUN/PARAM-Funktionen,

ⓒ die durch Standardbelegung in Kraft befindlichen COBRUN/PARAM-Funktionen zum Zeitpunkt der Übersetzung und

ⓓ Informationen für Wartungs- und Diagnosezwecke.

```
  Ⓐ              Ⓑ                       Ⓓ                                      Ⓔ          Ⓕ          Ⓖ
  COB1 V02.3A    COBOL-74 COMPILATION    COBRUN/PARAM LISTING                   15:31:09   86-05-27   PAGE 0001
ⓐ ENVIRONMENT
  -----------

                          PROCESSOR          :   7.571

                          OPERATING SYSTEM   :   BS2000 V8.0

                          COMPILER           :   COB1 V02.3A    COBOL-74


ⓑ OPTIONS IN EFFECT
  -----------------

                          DIAGTEXT=GERMAN
                          DIAG=YES
                          ERRLST=NO
                          LIST=YES
                          MAP=YES
                          MODULE=PLAM.LIB
                          OBJLST=YES
                          SYMDIC=YES
                          WRLST
                          XREF=YES


ⓒ OPTIONS BY DEFAULT
  ------------------

                          ACTKEY=TTTR
                          ERRPRI
                          LINE64
                          QUOTE="
                          RANGECHECK=NO
                          SAVLST=NO
                          TRUNCATE=YES
                          SYMTEST=MAP


ⓓ REV# = A
  (FOR INTERNAL USE)
```

2.3.4.2 Quellprogrammliste

Die von COB1 erzeugte Quellprogrammliste protokolliert

- alle eingegebenen Quellprogrammsätze einschließlich der durch eine COPY-Anweisung hineinkopierten Quellprogrammteile.
- alle zur Auflösung vorhandener COPY's benötigten Bibliotheken.

Durch den COBRUN-Operanden LINEnn kann der Seitenwechsel für die Übersetzungslisten beeinflußt werden. COB1 erzeugt standardmäßig nach 64 Zeilen einen Seitenwechsel. Der mit nn angegebene Wert legt die neue maximale Zeilenanzahl je Seite fest.

Eine weitere Möglichkeit, das Listenbild zu beeinflussen, stellt eine Quellprogrammanweisung mit dem Zeichen "/" in Spalte 7 dar ②. Eine solche Anweisung kann in jeder beliebigen Zeile des Quellprogramms auftreten und bewirkt ebenfalls einen Seitenvorschub.

Durch folgende COBRUN-Operanden kann die Protokollierung spezieller Quellcodeteile unterdrückt werden.

- COBRUN NOCOPY
  Die durch COPY-Anweisungen eingefügten Quellcodeteile werden nicht protokolliert. (Daneben kann in der einzelnen COPY-Anweisung durch den Zusatz SUPPRESS festgelegt werden, daß nur der Quellcode, der durch diese Anweisung eingefügt wird, nicht in die Quellprogrammliste aufgenommen wird (siehe Abschnitt 2.2.4).)
- COBRUN NODDLIST
  Die SUB-SCHEMA SECTION wird nicht protokolliert (siehe Anhang 4).

Zu Beginn jeder Seite einer Quellprogrammliste wird nach der Überschrift eine Zeile erzeugt, die Spaltenmarkierungen ① enthält. Diese Markierungen entsprechen dem COBOL-Format und erleichtern es dem Benutzer, Mißachtung des von COBOL geforderten Spaltenformats zu erkennen.

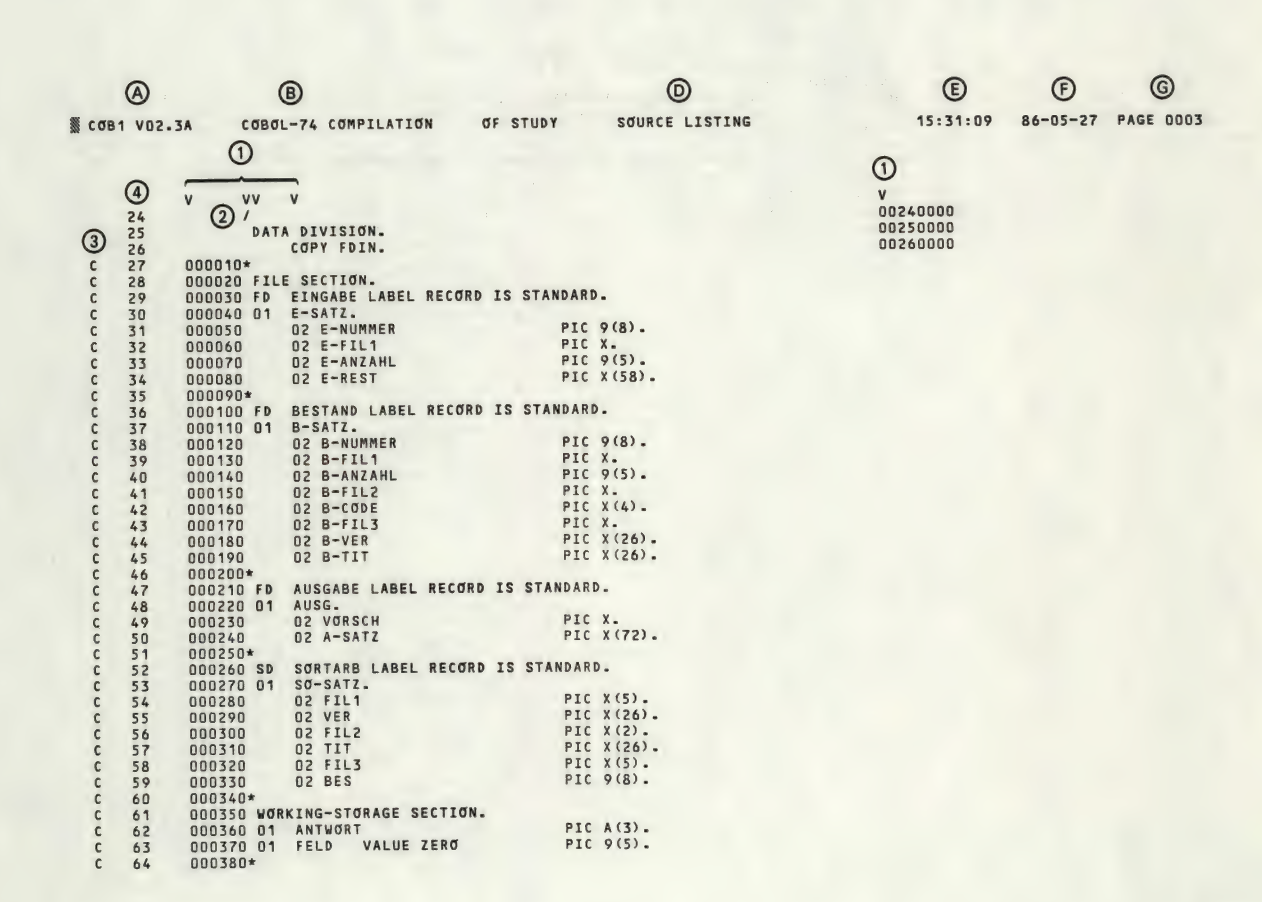
Jede Zeile einer Quellprogrammliste ist in die folgenden Bereiche unterteilt:

Anzeigenfeld	(Spalte 1) ③ informiert den Benutzer über Fehler innerhalb der benutzereigenen Numerierung der Eingabesätze (Anzeige S) und über Verstöße gegen die maximale Zeilenlänge von 80 Zeichen (Anzeige T). Außerdem werden in ihm Sätze gekennzeichnet, die aus einer COPY-Bibliothek kopiert wurden (Anzeige C) oder die zur SUB-SCHEMA-SECTION gehören (Anzeige D).
Folgenummernfeld	(Spalte 2—6) ④ enthält eine von COB1 vergebene, maximal 5-stellige Nummer, die zur Kennzeichnung des eingegebenen Quellprogrammsatzes dient. Diese Nummer dient zur eindeutigen Identifizierung der Quellcodezeilen. Sie findet sich in allen von COB1 erzeugten Listen als Querverweisnummer wieder und wird zur Verknüpfung mit etwaigen Fehlermeldungen verwendet. Der maximale Wert beträgt 32767. Überschreitet ein Quellprogramm diese Zahl, wird wieder von 0 an numeriert.
Quellprogrammbereich	(Spalte 11—90) enthält den vom Benutzer eingegebenen Satz. Dabei ist zu beachten, daß nur abdruckbare Zeichen dargestellt werden.

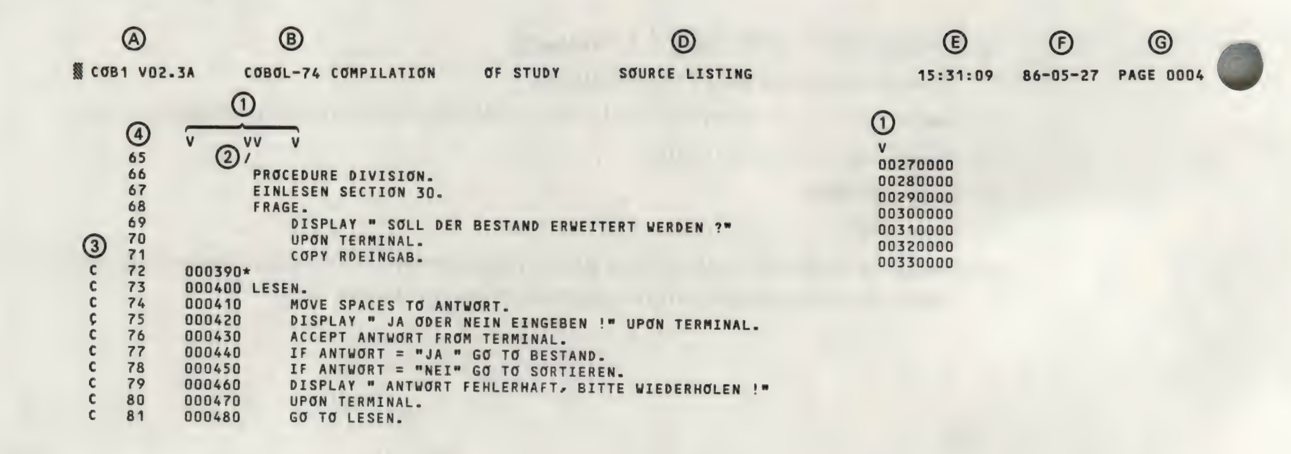
Als letzte Seite der Quellprogrammliste wird eine Bibliotheksliste ausgegeben. Ihr sind die Quellen zu entnehmen, aus denen das in dieser Übersetzung bearbeitete COBOL-Programm entstand. In der ersten Zeile steht, daß Source über SYSDTA eingelesen wurde. Weitere, je COPY-Anweisung angelegte Zeilen enthalten folgende Informationen.

- ⓐ Folgenummer ④ der COPY-Anweisung
- ⓑ Linkname aus der COPY-Anweisung
- ⓒ Dateiname, unter dem die Bibliothek im BS2000-Dateikatalog eingetragen ist
- ⓓ Bibliothekstyp (LMS, PLAM)
- ⓔ Elementname
- ⓕ Datum und
- ⓖ Versions-Nummer, mit der das Bibliothekselement in der Bibliothek eingetragen ist. Datum und Versions-Nummer sind nicht immer vorhanden.

```
   Ⓐ              Ⓑ                                  Ⓓ                     Ⓔ          Ⓕ          Ⓖ
 COB1 V02.3A    COBOL-74 COMPILATION               SOURCE LISTING        15:31:09   86-05-27   PAGE 0002
                   ①
 ④   V        VV   V                                           ①
                                                               V
  1            IDENTIFICATION DIVISION.                        00010000
  2            PROGRAM-ID.               STUDY.                00020000
  3            AUTHOR.                   RUDOLF.               00030000
  4           *                                                00040000
  5            ENVIRONMENT DIVISION.                           00050000
  6            CONFIGURATION SECTION.                          00060000
  7            OBJECT-COMPUTER.                                00070000
  8                SEGMENT-LIMIT IS 20.                        00080000
  9            INPUT-OUTPUT SECTION.                           00090000
 10            FILE-CONTROL.                                   00100000
 11                SELECT EINGABE ASSIGN TO DA-590-I-SYS010    00110000
 12                ORGANIZATION IS INDEXED                     00120000
 13                ACCESS MODE IS DYNAMIC                      00130000
 14                RECORD KEY IS E-NUMMER.                     00140000
 15           *                                                00150000
 16                SELECT BESTAND ASSIGN TO DA-590-I-SYS010    00160000
 17                ORGANIZATION IS INDEXED                     00170000
 18                ACCESS MODE IS DYNAMIC                      00180000
 19                RECORD KEY IS B-NUMMER.                     00190000
 20           *                                                00200000
 21                SELECT AUSGABE ASSIGN TO UR-PRINTERDOD-S-SYS020.  00210000
 22           *                                                00220000
 23                SELECT SORTARB ASSIGN TO DISC.              00230000
```

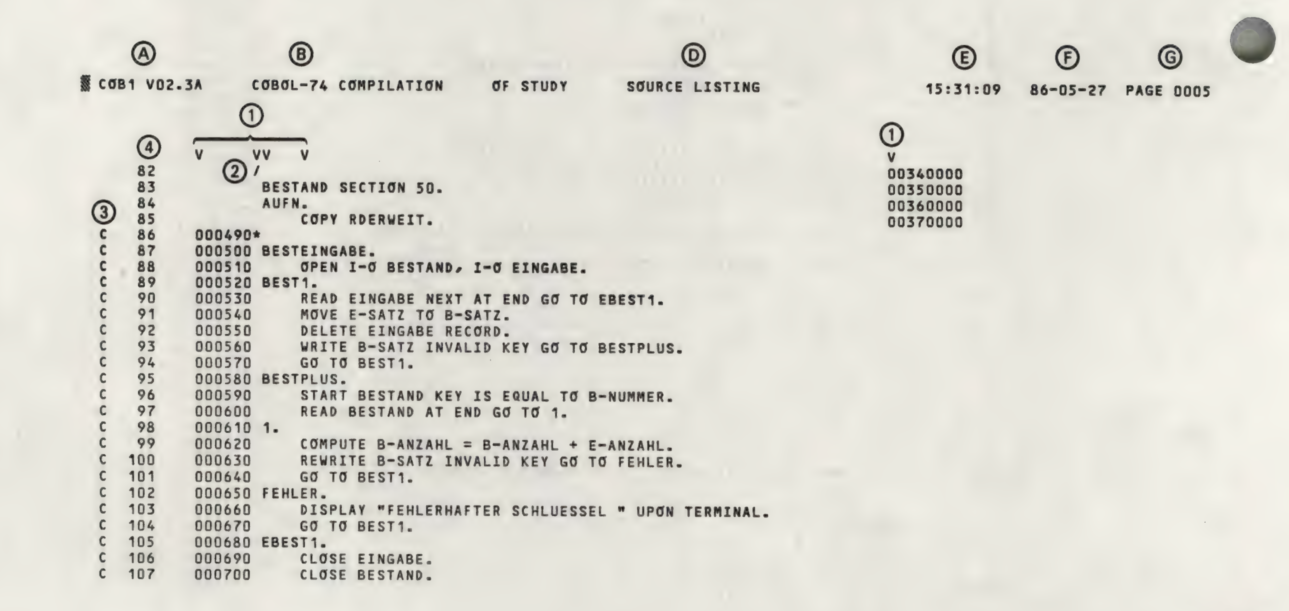


```
   Ⓐ              Ⓑ                                          Ⓓ                  Ⓔ          Ⓕ          Ⓖ
 COB1 V02.3A    COBOL-74 COMPILATION       OF STUDY      SOURCE LISTING     15:31:09   86-05-27   PAGE 0003
                        ①
      ④      V        VV   V                                                ①
                                                                            V
      24       ② /                                                          00240000
      25           DATA DIVISION.                                           00250000
 ③    26               COPY FDIN.                                           00260000
 C    27     000010*
 C    28     000020 FILE SECTION.
 C    29     000030 FD  EINGABE LABEL RECORD IS STANDARD.
 C    30     000040 01  E-SATZ.
 C    31     000050     02 E-NUMMER              PIC 9(8).
 C    32     000060     02 E-FIL1                PIC X.
 C    33     000070     02 E-ANZAHL              PIC 9(5).
 C    34     000080     02 E-REST                PIC X(58).
 C    35     000090*
 C    36     000100 FD  BESTAND LABEL RECORD IS STANDARD.
 C    37     000110 01  B-SATZ.
 C    38     000120     02 B-NUMMER              PIC 9(8).
 C    39     000130     02 B-FIL1                PIC X.
 C    40     000140     02 B-ANZAHL              PIC 9(5).
 C    41     000150     02 B-FIL2                PIC X.
 C    42     000160     02 B-CODE                PIC X(4).
 C    43     000170     02 B-FIL3                PIC X.
 C    44     000180     02 B-VER                 PIC X(26).
 C    45     000190     02 B-TIT                 PIC X(26).
 C    46     000200*
 C    47     000210 FD  AUSGABE LABEL RECORD IS STANDARD.
 C    48     000220 01  AUSG.
 C    49     000230     02 VORSCH                PIC X.
 C    50     000240     02 A-SATZ                PIC X(72).
 C    51     000250*
 C    52     000260 SD  SORTARB LABEL RECORD IS STANDARD.
 C    53     000270 01  SO-SATZ.
 C    54     000280     02 FIL1                  PIC X(5).
 C    55     000290     02 VER                   PIC X(26).
 C    56     000300     02 FIL2                  PIC X(2).
 C    57     000310     02 TIT                   PIC X(26).
 C    58     000320     02 FIL3                  PIC X(5).
 C    59     000330     02 BES                   PIC 9(8).
 C    60     000340*
 C    61     000350 WORKING-STORAGE SECTION.
 C    62     000360 01  ANTWORT                  PIC A(3).
 C    63     000370 01  FELD   VALUE ZERO        PIC 9(5).
 C    64     000380*
```



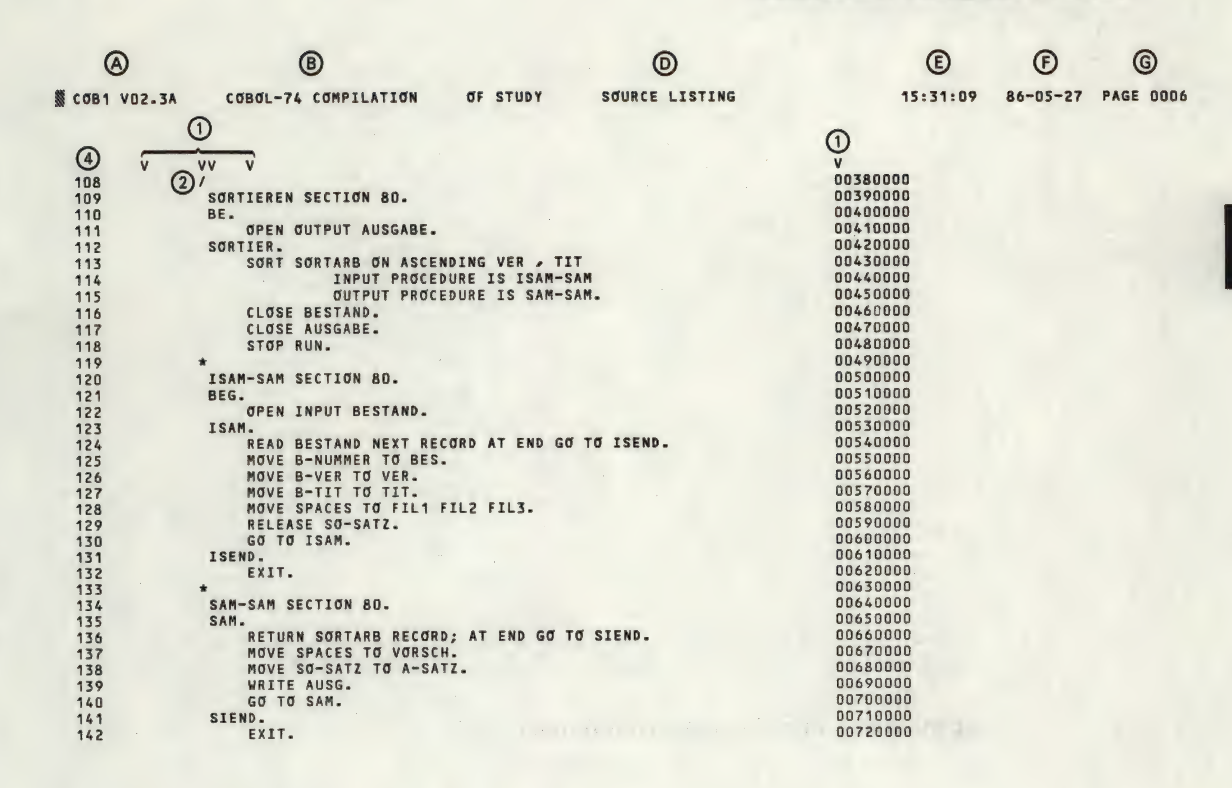
```
COB1 V02.3A   COBOL-74 COMPILATION   OF STUDY   SOURCE LISTING          15:31:09   86-05-27   PAGE 0004

        V      VV    V                                                  V
    65          /                                                       00270000
    66           PROCEDURE DIVISION.                                    00280000
    67           EINLESEN SECTION 30.                                   00290000
    68           FRAGE.                                                 00300000
    69               DISPLAY " SOLL DER BESTAND ERWEITERT WERDEN ?"     00310000
    70               UPON TERMINAL.                                     00320000
    71               COPY RDEINGAB.                                     00330000
C   72    000390*
C   73    000400 LESEN.
C   74    000410     MOVE SPACES TO ANTWORT.
C   75    000420     DISPLAY " JA ODER NEIN EINGEBEN !" UPON TERMINAL.
C   76    000430     ACCEPT ANTWORT FROM TERMINAL.
C   77    000440     IF ANTWORT = "JA " GO TO BESTAND.
C   78    000450     IF ANTWORT = "NEI" GO TO SORTIEREN.
C   79    000460     DISPLAY " ANTWORT FEHLERHAFT, BITTE WIEDERHOLEN !"
C   80    000470     UPON TERMINAL.
C   81    000480     GO TO LESEN.
```



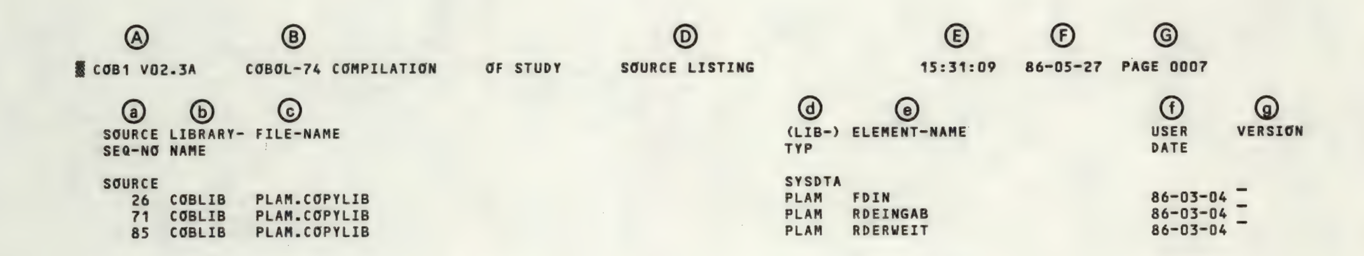
```
COB1 V02.3A   COBOL-74 COMPILATION   OF STUDY   SOURCE LISTING          15:31:09   86-05-27   PAGE 0005

        V      VV    V                                                  V
    82          /                                                       00340000
    83           BESTAND SECTION 50.                                    00350000
    84           AUFN.                                                  00360000
    85               COPY RDERWEIT.                                     00370000
C   86    000490*
C   87    000500 BESTEINGABE.
C   88    000510     OPEN I-O BESTAND, I-O EINGABE.
C   89    000520 BEST1.
C   90    000530     READ EINGABE NEXT AT END GO TO EBEST1.
C   91    000540     MOVE E-SATZ TO B-SATZ.
C   92    000550     DELETE EINGABE RECORD.
C   93    000560     WRITE B-SATZ INVALID KEY GO TO BESTPLUS.
C   94    000570     GO TO BEST1.
C   95    000580 BESTPLUS.
C   96    000590     START BESTAND KEY IS EQUAL TO B-NUMMER.
C   97    000600     READ BESTAND AT END GO TO 1.
C   98    000610 1.
C   99    000620     COMPUTE B-ANZAHL = B-ANZAHL + E-ANZAHL.
C  100    000630     REWRITE B-SATZ INVALID KEY GO TO FEHLER.
C  101    000640     GO TO BEST1.
C  102    000650 FEHLER.
C  103    000660     DISPLAY "FEHLERHAFTER SCHLUESSEL " UPON TERMINAL.
C  104    000670     GO TO BEST1.
C  105    000680 EBEST1.
C  106    000690     CLOSE EINGABE.
C  107    000700     CLOSE BESTAND.
```



```
      (A)              (B)                          (D)                      (E)        (F)       (G)
COB1 V02.3A     COBOL-74 COMPILATION     OF STUDY     SOURCE LISTING          15:31:09   86-05-27  PAGE 0006

              (1)                                                       (1)
 (4)   V      VV   V                                                     V
 108          (2)/                                                      00380000
 109          SORTIEREN SECTION 80.                                     00390000
 110          BE.                                                       00400000
 111              OPEN OUTPUT AUSGABE.                                  00410000
 112          SORTIER.                                                  00420000
 113              SORT SORTARB ON ASCENDING VER , TIT                   00430000
 114                       INPUT PROCEDURE IS ISAM-SAM                  00440000
 115                       OUTPUT PROCEDURE IS SAM-SAM.                 00450000
 116              CLOSE BESTAND.                                        00460000
 117              CLOSE AUSGABE.                                        00470000
 118              STOP RUN.                                             00480000
 119         *                                                          00490000
 120          ISAM-SAM SECTION 80.                                      00500000
 121          BEG.                                                      00510000
 122              OPEN INPUT BESTAND.                                   00520000
 123          ISAM.                                                     00530000
 124              READ BESTAND NEXT RECORD AT END GO TO ISEND.          00540000
 125              MOVE B-NUMMER TO BES.                                 00550000
 126              MOVE B-VER TO VER.                                    00560000
 127              MOVE B-TIT TO TIT.                                    00570000
 128              MOVE SPACES TO FIL1 FIL2 FIL3.                        00580000
 129              RELEASE SO-SATZ.                                      00590000
 130              GO TO ISAM.                                           00600000
 131          ISEND.                                                    00610000
 132              EXIT.                                                 00620000
 133         *                                                          00630000
 134          SAM-SAM SECTION 80.                                       00640000
 135          SAM.                                                      00650000
 136              RETURN SORTARB RECORD; AT END GO TO SIEND.            00660000
 137              MOVE SPACES TO VORSCH.                                00670000
 138              MOVE SO-SATZ TO A-SATZ.                               00680000
 139              WRITE AUSG.                                           00690000
 140              GO TO SAM.                                            00700000
 141          SIEND.                                                    00710000
 142              EXIT.                                                 00720000
```



```
      (A)              (B)                          (D)                      (E)        (F)       (G)
COB1 V02.3A     COBOL-74 COMPILATION     OF STUDY     SOURCE LISTING          15:31:09   86-05-27  PAGE 0007

   (a)     (b)       (c)                                       (d)      (e)                        (f)       (g)
 SOURCE  LIBRARY-  FILE-NAME                                  (LIB-)  ELEMENT-NAME                USER      VERSION
 SEQ-NO  NAME                                                 TYP                                 DATE

 SOURCE                                                       SYSDTA
     26  COBLIB    PLAM.COPYLIB                               PLAM    FDIN                        86-03-04  -
     71  COBLIB    PLAM.COPYLIB                               PLAM    RDEINGAB                    86-03-04  -
     85  COBLIB    PLAM.COPYLIB                               PLAM    RDERWEIT                    86-03-04  -
```

Leerseite durch den Nachtrag vom August 1986

2.3.4.3 Objektprogrammliste

Die Objektprogrammliste wird nur auf Anforderung durch den Benutzer ausgegeben. Sie muß mit dem Kommando

```
/PARAM OBJLST=YES
```

vor dem Aufruf des COB1-Übersetzers angefordert werden.

Das Listenbild der Objektprogrammliste kann vom Benutzer durch den COBRUN-Operanden LINExx beeinflußt werden.

Die Objektprogrammliste enthält Informationen über die erzeugten Datenbereiche, Kontrollblöcke und Maschinenbefehle. Außerdem werden die Namen der vom COB1-Übersetzer ausgewählten Ablaufzeitmoduln, die Namen der Einsprungpunkte und aufgerufenen Unterprogramme aufgelistet. Die Anordnung der einzelnen Teile der Objektprogrammliste entspricht der Speicherstruktur, die das Maschinenprogramm belegt.

Die Objektprogrammliste läßt sich in vier wesentliche Abschnitte gliedern:

Abschnitt 1 der Liste informiert den Benutzer über die verwendeten externen Namen, die vom COB1-Übersetzer standardmäßig generierten Unterprogramme sowie über notwendige Tabellen und Speicherbereiche. Außerdem enthält dieser Abschnitt die Steuerblöcke, die zum Anschluß an Betriebssystemkomponenten wie DVS und UDS notwendig sind.

Abschnitt 2 der Liste enthält eine symbolische Darstellung der Konstanten und Arbeitsbereiche aus dem Datenteil des Programms. Außerdem werden alle standardmäßig generierten Konstanten und die Literale des Prozedurteils aufgeführt.

Abschnitt 3 enthält eine symbolische Darstellung aller vom COB1-Übersetzer erzeugten Maschinenbefehle. Zusätzlich werden Hinweise auf Paragraphen und Kapitelnamen, Quellprogrammfolgenummern und die ursprüngliche Quellprogrammanweisung gegeben.

Abschnitt 4 wird nur erstellt, wenn die PARAM-Funktion SYMDIC=YES wirkt (siehe Abschnitt 2.4.2). Er enthält dann das interne symbolische Adreßbuch (ISD).

```
 COB1 V02.3A      CØBØL-74 CØMPILATIØN     STUDY          ØBJECT PRØGRAM LISTING         15:31:09    86-05-27  PAGE 0008

                                                        EXTERNAL SYMBØL DICTIØNARY

                                REL.  LENGTH  LENGTH
 SYMBØL     TYPE ID        ADDR.   HEX.    DEC.       REMARKS


                                                      ***** CØMPILATIØN UNIT ******
 STUDY        CU 0000

                                                      ***** CØNTRØL SECTIØNS *****
 STUDY        SD 0001   000000  001345 0004933

                                                      ***** SUBRØUTINE ENTRY PØINTS *****
 ITCOPØVH     ER 0002
 ITCMSIN1     ER 0003
 ITCMIIN1     ER 0004
 ITCOSEGO     ER 0005
 ITCOENDO     ER 0006
 ITCMSØPO     ER 0007
 ITCMSWRO     ER 0008
 ITCMSCLO     ER 0009
 ITCMIØPO     ER 000A
 ITCMIRDO     ER 000B
 ITCMIWRO     ER 000C
 ITCMICLO     ER 000D
 ITCOSMGO     ER 000E
 ITCOACAO     ER 000F
 ITCODSAO     ER 0010
 ITCXINTO     ER 0011

                                                      ***** ENTRY-NAMES IN CALL-SØURCE-STATEMENTS ØR SEGMENT-NAMES *****
 STUDY30      VC 0012
 STUDY50      VC 0013
 STUDY80      VC 0014
 STUDY30      ER 0015
 STUDY50      ER 0016
 STUDY80      ER 0017
```

```
 COB1 V02.3A      CØBØL-74 CØMPILATIØN     STUDY          ØBJECT PRØGRAM LISTING         15:31:09    86-05-27  PAGE 0009

                                                        ØBJECT PRØGRAM TABLES CØDE

  REL    TYPE
 LØCTN   INST   ØBJECT CØDE          REMARKS                                               GRAPHICS


                                     ***** INITIALIZATIØN CØDING (PART 1) *****
 000000  BALR   05 F0
 000002  LM     98 2D F00A
 000006  BC     47 F0 F03A
 00000A  BCR    07 00

                                     ***** CØBØL REGISTER INITIAL VALUES *****
 00000C  DC-X   00
 00000D  DC-A   001130               ADDR CØNST PRØC. LITERALS
 000010  DC-A   00001050             ADDR CØNST ASSIGNED PERM REGR 3
 000014  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR 4
 000018  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR 5
 00001C  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR 6
 000020  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR 7
 000024  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR 8
 000028  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR 9
 00002C  DC-X   00000000             ADDR CØNST ASSIGNED PERM REGR A
 000030  DC-A   00000050             ADDR CØNST ASSIGNED PERM REGR B
 000034  DC-A   00000000             REGR C PERM ASSIGNED TØ SR ITCOPØVH
 000038  DC-A   00001280             ADDR 1ST EXEC. SECTIØN/PARAGRAPH

                                     ***** INITIALIZATIØN CØDING (PART 2) *****
 00003C  BAL    45 E0 C0C8
 000040  SLR    1F EE
 000042  SPM    04 E0
 000044  L      58 F0 B078
 000048  BALR   05 EF
 00004A  BCR    07 FD
 00004C  DS

                                     ***** GØ TØ RØØT --- RØØT ØR WITHIN SEGMENT *****
 000050  LH     48 F0 E002
 000054  L      58 FF B000
 000058  BCR    07 FF

                                     ***** ABNØRMAL TERMINATIØN *****
 00005A  BC     47 F0 C0DC

                                     ***** VERSIØN ØF LC - INTERFACE *****
 00005E  DC-X   0001

                                     ***** SPECIAL A-CØNSTANTS *****
 000060  DC-A   000000DC             FCB TABLE ADDRESS
 000064  DC-A   00000148             PERFØRM EXIT BUCKET BASE ADDR
 000068  DC-A   0000005A             TERMD ADDRESS
 00006C  DC-A   00000158             ENTER PARAMETER ADDRESS
 000070  DC-A   00000158             SEGMENT PARAM BASE ADDR
 000074  DC-A   000000D4             SØRT/MERGE TABLE ADDRESS
 000078  DC-A   0000000C             PØINTER CØBØL REGISTERS
 00007C  DC-A   0000128E             PRØGRAM ID BLØCK ADDR
 000080  DC-X   00000000
 000084  DC-X   00000000

                                     ***** BAC ADDR CØNSTANTS *****
```

```
COB1 V02.3A      COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING          15:31:09    86-05-27  PAGE 0010

                                                          OBJECT PROGRAM TABLES CODE

  REL    TYPE
 LOCTN   INST   OBJECT CODE          REMARKS                                                       GRAPHICS

000088  DC-X   00000000
00008C  DC-A   00001050
000090  DC-A   00000F20
000094  DC-A   00000FB8
                                     ***** OBJECT SUBR ADDR CONSTANTS *****
000098  DC-A   00000000              ITCOSEGO
00009C  DC-A   00000000              ITCOENDO
0000A0  DC-A   00000000              ITCMSOPO
0000A4  DC-A   00000000              ITCMSWRO
0000A8  DC-A   00000000              ITCMSCLO
0000AC  DC-A   00000000              ITCMIOPO
0000B0  DC-A   00000000              ITCMIRDO
0000B4  DC-A   00000000              ITCMIWRO
0000B8  DC-A   00000000              ITCMICLO
0000BC  DC-A   00000000              ITCOSMGO
0000C0  DC-A   00000000              ITCOACAO
0000C4  DC-A   00000000              ITCODSAO
0000C8  DC-A   00000000              ITCXINTO
0000CC  DC-A   00000000              ITCMSIN1
0000D0  DC-A   00000000              ITCMIIN1
                                     ***** SORT/MERGE PARAMETERS ADDRESS CONSTANTS *****
0000D4  DC-A   00000F10
0000D8  DC-A   00000ED8
                                     ***** DTF ADDR CONSTANTS *****
0000DC  DC-A   000001F0
0000E0  DC-A   00000250
0000E4  DC-A   000006A0
0000E8  DC-A   00000700
0000EC  DC-A   00000B50
0000F0  DC-A   00000BD0
0000F4  DC-X   00000000
0000F8  DC-X   00000000
                                     ***** PROC. HEADER TAGS *****
0000FC  DC-A   00001280
000100  DC-A   00001280
000104  DC-A   00001280
000108  DC-X   81
000109  DC-X   000004
00010C  DC-X   81
00010D  DC-X   000018
000110  DC-X   82
000111  DC-X   000004
000114  DC-X   82
000115  DC-X   00001C
000118  DC-X   82
000119  DC-X   000074
00011C  DC-X   82
00011D  DC-X   0000A2
000120  DC-X   82
000121  DC-X   0000FA
000124  DC-X   82
```

```
COB1 V02.3A      COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING          15:31:09    86-05-27  PAGE 0011

                                                          OBJECT PROGRAM TABLES CODE

  REL    TYPE
 LOCTN   INST   OBJECT CODE          REMARKS                                                       GRAPHICS

000125  DC-X   000118
000128  DC-X   83
000129  DC-X   000004
00012C  DC-X   83
00012D  DC-X   000058
000130  DC-X   83
000131  DC-X   000064
000134  DC-X   83
000135  DC-X   0000C4
000138  DC-X   83
000139  DC-X   0000CA
00013C  DC-X   83
00013D  DC-X   0000CA
000140  DC-X   83
000141  DC-X   000112
000144  DC-X   0000
000146  DS
                                     ***** PERFORM EXIT BUCKETS *****
000148  DC-C   0000000000000000
000150  DC-C   0000000000000000
                                     ***** V-TYPE ADDRS FOR EACH NON-PERMANENT SEGMENT *****
000158  DC-A   00000000
00015C  DC-A   00000000
000160  DC-A   00000000
                                     ***** TEMPORARY SLICE ADDRESSES *****
000164  DS
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING          15:31:09    86-05-27  PAGE 0012

                                                             DATA DIVISION CODE

 REL   TYPE
LOCTN  CODE  OBJECT CODE            REMARKS                                                              GRAPHICS

000168 DS                           USE ERROR EXIT STACK

                                    ***** ******************* FILE:- EINGABE ************************ *****
000178 DC-X  00020074004C0050       FCB-GENERATION-PARAMETERS
             000000A0000000D8
             10050D0A00000000
000190 DC-A  00000F18
000194 DC-A  00000FB8
000198 DC-X  0000000000000000
             0008000400000000
             0000000007C5C9D5
             C7C1C2C540404040
             4040404040404040
             4040404040404040
             4040400200000000
             0000000000004800
             114E3000000400C5
             C9D5C7C1C2C54000
             00000000
0001EC DC-A  00000178
0001F0 DC-X  000100600000C900       COBOL INTERNAL FILE CONTROL BLOCK
             0000000000000000
             0000000001000000
             0000000000000000
             0000000400000000
             0000000000000048
             0000000800000048
             0000004880000000
             00000000
000234 DC-A  00001054
000238 DC-X  0000000400000000
             0000000000000000
000248 DS
000250 DS                           **** ISAM-FCB ****

                                    ***** ******************* FILE:- BESTAND ************************ *****
000628 DC-X  00020074004C0050       FCB-GENERATION-PARAMETERS
             000000A0000000D8
             10050D0A00000000
000640 DC-A  00000F18
000644 DC-A  00000FB8
000648 DC-X  0000000000000000
             0008000400000000
             0000000007C2C5E2
             E3C1D5C440404040
             4040404040404040
             4040404040404040
             4040400200000000
             0000000000004800
             114E3000000400C2
             C5E2E3C1D5C44000
             00000000
00069C DC-A  00000628
0006A0 DC-X  000100600000C900       COBOL INTERNAL FILE CONTROL BLOCK
             0000000000000000
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY         OBJECT PROGRAM LISTING        15:31:09   86-05-27  PAGE 0013

                                                             DATA DIVISION CODE

 REL    TYPE
LOCTN   CODE   OBJECT CODE          REMARKS                                                   GRAPHICS

               0000000002000000
               0000000000000000
               0000000400000000
               0000000000000048
               0000000800000048
               0000004880000000
               00000000
0006E4  DC-A   000010A4
0006E8  DC-X   0000000400000000
               0000000000000000
0006F8  DS
000700  DS                          **** ISAM-FCB ****

                                    ***** ****************** FILE:- AUSGABE *********************** *****
000AD8  DC-X   00020074004D0051     FCB-GENERATION-PARAMETERS
               00000060000000D8
               2203031400000000
000AF0  DC-A   00000F78
000AF4  DC-A   00000FD8
000AF8  DC-X   0000000000000000
               0000000000000000
               0000000007C1E4E2
               C7C1C2C540404040
               4040404040404040
               4040404040404040
               4040400200000000
               0000000000004900
               118A0401000400C1
               E4E2C7C1C2C54000
               00000000
000B4C  DC-A   00000AD8
000B50  DC-X   000100800000E200     COBOL INTERNAL FILE CONTROL BLOCK
               0000000000000000
               0000000003000000
               0000000000000000
               0000000400000000
               0000000000000000
               0000000000000049
               0000000100000049
               00000049
000B94  DC-S   0040
000B96  DC-X   9000000000000000
               0000
000BA0  DC-A   00000F20
000BA4  DC-X   0000000000000000
               0000000000040000
               0000000000000000
               0000004D00000000
               0000000000000000
               00000000
000BD0  DS                          **** SAM-FCB ****
000ED0  DC-X   FFFFFFFFFFFFFFFF

                                    ***** ****************** CONTROL BLOCK FOR SORT/MERGE NO. 01 AT SEQ. NO. 00113 *****
000ED8  DC-A   00000FB8
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY         OBJECT PROGRAM LISTING        15:31:09   86-05-27  PAGE 0014

                                                             DATA DIVISION CODE

 REL    TYPE
LOCTN   CODE   OBJECT CODE          REMARKS                                                   GRAPHICS

000EDC  DC-X   0000000000480034
               0000000000000000
               0000000000000000
               00010071C0000708
               0900000000000200
               0005001A01040021
               001A0104
                                    ***** INDEPENDENT DATA AREA *****
000F10  DS
                                    ***** FIXED DATA AREA *****
001050  DS
0010F8  DC-C   F0F0F0F0F0                                                                     00000
0010FD  DS
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING         15:31:09    86-05-27  PAGE 0015

                                                 PROCEDURE DIVISION CONSTANTS

             REL    TYPE
            LOCTN   CODE   OBJECT CODE                                                                  GRAPHICS


            001130  DC-X   5B00000047F00000 47F0000000000000 4B41023A00040000 0000000000000000
                           F000000000000000 00000000
            00115C  DC-A   00000168
            001160  DC-A   00000168
            001164  DC-A   00000000
            001168  DC-A   00001000
            00116C  DC-A   00000168
            001170  DC-A   00000168
            001174  DC-X   00000000
            001178  DC-A   00000168
            00117C  DC-A   00000168
            001180  DC-A   00000168
            001184  DC-X   0000000000000000 0000000000000000 0000000000000000
            00119C  DC-A   00000168
            0011A0  DC-X   0000000000000000 0000000000000000 00000000000C0000 0000000000000000
            0011C0  DC-A   00001280             PERMANENT SLICE ADDRESSES
            0011C4  DC-A   00000000
            0011C8  DC-A   00000000
            0011CC  DC-A   00000000
            0011D0  DS
            001200  DC-C   40E2D6D3D340C4C5 D940C2C5E2E3C1D5 C440C5D9E6C5C9E3 C5D9E340E6C5D9C4    SOLL DER BESTAND ERWEITERT WERD
                           C5D5406F40D1C140 D6C4C5D940D5C5C9 D540C5C9D5C7C5C2 C5D5405A40C1D5E3   EN ? JA ODER NEIN EINGEBEN ! ANT
                           E6D6D9E340C6C5C8 D3C5D9C8C1C6E36B 40C2C9E3E3C540E6 C9C5C4C5D9C8D6D3   WORT FEHLERHAFT, BITTE WIEDERHOL
                           C5D5405AF1F2C6C5 C8D3C5D9C8C1C6E3 C5D940E2C3C8D3E4 C5E2E2C5D34000     EN !12FEHLERHAFTER SCHLUESSEL
            00127F  DS
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING         15:31:09    86-05-27  PAGE 0016

                                                 PROCEDURE DIVISION CODE

SOURCE  REL     TYPE                    EFF     EFF                          SOURCE  REL    TYPE                  EFF     EFF
SEQ.NO. LOCTN   INST   OBJECT CODE      ADDR1   ADDR2     REMARKS            SEQ.NO. LOCTN  INST  OBJECT CODE     ADDR1   ADDR2     REMARKS


00066   001280  BAL    45E0C0B2         0000B2            GO TO
        001284  DC-S   00B8
        001286  L      58F0C10C         00010C
        00128A  BAL    45EF0008
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING         15:31:09    86-05-27  PAGE 0017

                   PROGRAM IDENTIFICATION, COMPILE TIME INFORMATION, LIBRARY MODULE AND PROGRAM STORAGE SUMMARY


            00128E  DC-C   E2E3E4C4E8404040 40F0F0F140C3D6D4 D7C9D3C5C440C2E8 40C3D6C2F1404040   STUDY    001 COMPILED BY COB1
                           E5C5D9E2C9D6D540 F24BF3C140404060 D9C5E57B40C14040 4040404040404040   VERSION 2.3A   -REV# A
                           4040404040404040 4040404040404040 4040404040404040 4040404040404040
                           40404040C24040 4040404040404040 4040404040404040 4040404040404040         B
                           4040404040404040 4040404040404040 4040404040604040 D6D540F8F660F0F5                     -  ON 86-05
                           60F2F740C1E340F1 F57AF3F17AF0F986 05271531093145                     -27 AT 15:31:09F.......
```

```
COB1 V02.3A      COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING          15:31:09    86-05-27  PAGE 0018

                                                   INTERNAL SYMBOL DICTIONARY


              EINGABE                                   E-SATZ                                   E-NUMMER
              E-FIL1                                    E-ANZAHL                                 E-REST
              BESTAND                                   B-SATZ                                   B-NUMMER
              B-FIL1                                    B-ANZAHL                                 B-FIL2
              B-CODE                                    B-FIL3                                   B-VER
              B-TIT                                     AUSGABE                                  AUSG
              VORSCH                                    A-SATZ                                   SORTARB
              SO-SATZ                                   FIL1                                     VER
              FIL2                                      TIT                                      FIL3
              BES                                       ANTWORT                                  FELD
              TALLY                                     SORT-FILE-SIZE                           SORT-CORE-SIZE
              SORT-MODE-SIZE                            SORT-RETURN                              $66GOT
         STUDY30
```



```
COB1 V02.3A      COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING          15:31:09    86-05-27  PAGE 0019

                                                   PROCEDURE DIVISION CODE

SOURCE   REL     TYPE                    EFF    EFF                        SOURCE   REL     TYPE                    EFF    EFF
SEQ.NO.  LOCTN   INST  OBJECT CODE       ADDR1  ADDR2    REMARKS           SEQ.NO.  LOCTN   INST  OBJECT CODE       ADDR1  ADDR2    REMARKS


***** EINLESEN 30 SECTION. *****

00067  000000  BCR   07F1
       000002  DC-S  00B8

***** FRAGE 30 PARAGRAPH. *****

00069  000004  MVC   D22330B520D0 001105 001200   DISPLAY
       00000A  LA    411030B5     001105
       00000E  L     58F0B074     0000C4
       000012  BALR  05EF
       000014  DC-X  04000024

***** LESEN 30 PARAGRAPH. *****

00074  000018  MVC   D20230A0C12E 0010F0 00012E   MOVE
00075  00001E  MVC   D21730B520F4 001105 001224   DISPLAY
       000024  LA    411030B5     001105
       000028  L     58F0B074     0000C4
       00002C  LR    1800
       00002E  BALR  05EF
       000030  DC-X  04000018
00076  000034  LA    411030B5     001105          ACCEPT
       000038  L     58F0B070     0000C0
       00003C  LR    1800
       00003E  BALR  05EF
       000040  DC-X  10000003
       000044  MVC   D20230A030B5 0010F0 001105
00077  00004A  CLC   D50230A020F5 0010F0 001225   IF
       000050  L     58D02094     0011C4
       000054  BC    4770D05E     00005E
00077  000058  BAL   45E0C0B2     0000B2          GO TO
       00005C  DC-S  00C0
00078  00005E  CLC   D50230A020FD 0010F0 00122D   IF
       000064  BC    4770D06E     00006E
00078  000068  BAL   45E0C0B2     0000B2          GO TO
       00006C  DC-S  00D8
00079  00006E  MVC   D22730B5210C 001105 00123C   DISPLAY
       000074  LA    411030B5     001105
       000078  L     58F0B074     0000C4
       00007C  LR    1800
       00007E  BALR  05EF
       000080  DC-X  04000028
00081  000084  BC    47F0D018     000018          GO TO
00082  000088  BAL   45E0C0B2     0000B2          GO TO
       00008C  DC-S  00C0
       00008E  L     58F0C10C     00010C
       000092  BAL   45EF0008
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING        15:31:09   86-05-27  PAGE 0020

                  PROGRAM IDENTIFICATION, COMPILE TIME INFORMATION, LIBRARY MODULE AND PROGRAM STORAGE SUMMARY


        000096  DC-C  E2E3E4C4E8F3F040 40F0F0F140C3D6D4 D7C9D3C5C440C2E8 40C3D6C2F1404040   STUDY30  001 COMPILED BY COB1
                      E5C5D9E2C9D6D540 F24BF3C140404060 D9C5E57B40C14040 4040404040404040   VERSION 2.3A   -REV# A
                      4040404040404040 4040404040404040 4040404040404040 4040404040404040
                      404040404C024040 4040404040404040 4040404040404040 4040404040404040        B
                      4040404040404040 4040404040404040 4040404040604040 D6D540F8F660F0F5                       -  ON 86-05
                      60F2F740C1E340F1 F57AF3F17AF0F986 05271531093145                       -27 AT 15:31:09F.......
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING        15:31:09   86-05-27  PAGE 0021

                                        INTERNAL SYMBOL DICTIONARY


        EINLESEN                              $69DIS                               FRAGE
        $74MOV                                LESEN                                $75DIS
        $76ACC                                $77IF                                $77GOT
        $78IF                                 $78GOT                               $79DIS
        $81GOT                                $81GOT2
    STUDY50
```

```
COB1 V02.3A     CØBØL-74 CØMPILATIØN    STUDY         ØBJECT PRØGRAM LISTING        15:31:09    86-05-27  PAGE 0022

                                                      PRØCEDURE DIVISIØN CØDE

SØURCE  REL     TYPE                 EFF.   EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1  ADDR2     REMARKS

***** BESTAND 50 SECTIØN. *****

00083   000000  BCR   07F1
        000002  DC-S  00C0

***** AUFN 50 PARAGRAPH. *****

***** BESTEINGABE 50 PARAGRAPH. *****

00088   000004  LM    9801B094       0000E4           ØPEN
        000008  L     58F0B05C       0000AC
        00000C  BALR  05EF
        00000E  DC-S  0208
00088   000010  LM    9801B08C       0000DC           ØPEN
        000014  L     58F0B05C       0000AC
        000018  BALR  05EF
        00001A  DC-S  0108

***** BEST1 50 PARAGRAPH. *****

00090   00001C  LM    9801B08C       0000DC           READ
        000020  L     58F0B060       0000B0
        000024  BALR  05EF
        000026  DC-S  0000
        000028  DC-S  8000
00090   00002A  CLC   D50020202134   001150 001264    READ
        000030  L     58D02098       0011C8
        000034  BC    4770D03C       00003C
00090   000038  BC    47F0D118       000118           GØ TØ
00091   00003C  MVC   D24730583008   0010A8 001058    MØVE
00092   000042  LM    9801B08C       0000DC           DELETE
        000046  L     58F0B064       0000B4
        00004A  BALR  05EF
        00004C  DC-S  4014           000014
        00004E  DC-S  0000
        000050  DC-S  0000
00093   000052  LM    9801B094       0000E4           WRITE
        000056  L     58F0B064       0000B4
        00005A  BALR  05EF
        00005C  DC-S  000C
        00005E  DC-S  0048
        000060  DC-S  8000
00093   000062  CLC   D50020202135   001150 001265    WRITE
        000068  BC    4770D070       000070
00093   00006C  BC    47F0D074       000074           GØ TØ
00094   000070  BC    47F0D01C       00001C           GØ TØ

***** BESTPLUS 50 PARAGRAPH. *****

00096   000074  LM    9801B094       0000E4           START
        000078  L     58F0B060       0000B0
        00007C  BALR  05EF
        00007E  DC-S  0008

SØURCE  REL     TYPE                 EFF    EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1  ADDR2     REMARKS

        000080  DC-S  0007
00097   000082  LM    9801B094       0000E4           READ
        000086  L     58F0B060       0000B0
        00008A  BALR  05EF
        00008C  DC-S  0000
        00008E  DC-S  8000
00097   000090  CLC   D50020202134   001150 001264    READ
        000096  L     58D02098       0011C8
        00009A  BC    4770D0A2       0000A2
00097   00009E  BC    47F0D0A2       0000A2           GØ TØ

***** 1 50 PARAGRAPH. *****

00099   0000A2  PACK  F234C2E03061   0002E0 0010B1    CØMPUTE
        0000A8  ØI    960FC2E3       0002E3
        0000AC  NI    94FCC2E3       0002E3
        0000B0  PACK  F234C3603011   000360 001061
        0000B6  ØI    960FC363       000363
        0000BA  NI    94FCC363       000363
        0000BE  AP    FA33C2E0C360   0002E0 000360
        0000C4  ZAP   F823C2B0C2E0   0002B0 0002E0
        0000CA  ØI    960FC2B2       0002B2
        0000CE  UNPK  F3423061C2B0   0010B1 0002B0
00100   0000D4  LM    9801B094       0000E4           REWRITE
        0000D8  L     58F0B064       0000B4
        0000DC  BALR  05EF
        0000DE  DC-S  0010
        0000E0  DC-S  0048
        0000E2  DC-S  8000
00100   0000E4  CLC   D50020202135   001150 001265    REWRITE
        0000EA  L     58D02098       0011C8
        0000EE  BC    4770D0F6       0000F6
00100   0000F2  BC    47F0D0FA       0000FA           GØ TØ
00101   0000F6  BC    47F0D01C       00001C           GØ TØ

***** FEHLER 50 PARAGRAPH. *****

00103   0000FA  MVC   D21730B52136   001105 001266    DISPLAY
        000100  LA    411030B5       001105
        000104  L     58F0B074       0000C4
        000108  LR    1800
        00010A  BALR  05EF
        00010C  DC-X  04000018
00104   000110  L     58D02098       0011C8           GØ TØ
        000114  BC    47F0D01C       00001C

***** EBEST1 50 PARAGRAPH. *****

00106   000118  LM    9801B08C       0000DC           CLØSE
        00011C  L     58F0B068       0000B8
        000120  BALR  05EF
        000122  DC-S  1000
00107   000124  LM    9801B094       0000E4           CLØSE
        000128  L     58F0B068       0000B8
        00012C  BALR  05EF
```

```
COB1 V02.3A     CØBØL-74 CØMPILATIØN    STUDY         ØBJECT PRØGRAM LISTING        15:31:09    86-05-27  PAGE 0023

                                                      PRØCEDURE DIVISIØN CØDE

SØURCE  REL     TYPE                 EFF    EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1  ADDR2     REMARKS

        00012E  DC-S  1000
00108   000130  BAL   45E0C0B2       0000B2           GØ TØ
        000134  DC-S  00D8
        000136  L     58F0C10C       00010C
        00013A  BAL   45EF0008
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY         OBJECT PROGRAM LISTING         15:31:09   86-05-27  PAGE 0024

                PROGRAM IDENTIFICATION, COMPILE TIME INFORMATION, LIBRARY MODULE AND PROGRAM STORAGE SUMMARY


00013E  DC-C  E2E3E4C4E8F5F040 40F0F0F140C3D6D4 D7C9D3C5C440C2E8 40C3D6C2F1404040   STUDY50  001 COMPILED BY COB1
              E5C5D9E2C9D6D540 F24BF3C140404060 D9C5E57B40C14040 4040404040404040   VERSION 2.3A   -REV# A
              4040404040404040 4040404040404040 4040404040404040 4040404040404040
              404040404OC24040 4040404040404040 4040404040404040 4040404040404040        B
              4040404040404040 4040404040404040 4040404040604040 D6D540F8F660F0F5                         -  ON 86-05
              60F2F740C1E340F1 F57AF3F17AF0F986 05271531093145                      -27 AT 15:31:09F.......
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY         OBJECT PROGRAM LISTING         15:31:09   86-05-27  PAGE 0025

                                        INTERNAL SYMBOL DICTIONARY


         BESTAND                                $88OPE                                 AUFN
         BESTEINGABE                            $90REA                                 BEST1
         $90GOT                                 $91MOV                                 $92DEL
         $93WRI                                 $93GOT                                 $94GOT
         $96STA                                 BESTPLUS                               $97REA
         $97GOT                                 $99COM                                 1
         $100REW                                $100GOT                                $101GOT
         $103DIS                                FEHLER                                 $104GOT
         $106CLO                                EBEST1                                 $107CLO
         $107GOT
STUDY80
```

```
COB1 V02.3A     COBØL-74 CØMPILATIØN     STUDY          ØBJECT PRØGRAM LISTING          15:31:09   86-05-27  PAGE 0026

                                                   PRØCEDURE DIVISIØN CØDE

SØURCE  REL     TYPE                 EFF     EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1   ADDR2     REMARKS

***** SØRTIEREN 80 SECTIØN. *****

00109   000000  BCR   07F1
        000002  DC-S  00D8

***** BE 80 PARAGRAPH. *****

00111   000004  LM    9801B09C       0000EC            ØPEN
        000008  L     58F0B050       0000A0
        00000C  BALR  05EF
        00000E  DC-S  0304
        000010  DC-S  0003

***** SØRTIER 80 PARAGRAPH. *****

00113   000012  LA    41000001                         SØRT
        000016  L     58F0B06C       0000BC
        00001A  BALR  05EF
00113   00001C  BALR  05EC                             SØRT
        00001E  DC-S  00DC
        000020  DC-S  0000
00113   000022  L     58F0B06C       0000BC            SØRT
        000026  BAL   45E0F004
00113   00002A  BALR  05EC                             SØRT
        00002C  DC-S  00E8
        00002E  DC-S  0008
00113   000030  L     58F0B06C       0000BC            SØRT
        000034  BAL   45E0F010
00116   000038  LM    9801B094       0000E4            CLØSE
        00003C  L     58F0B068       0000B8
        000040  BALR  05EF
        000042  DC-S  1000
00117   000044  LM    9801B09C       0000EC            CLØSE
        000048  L     58F0B058       0000A8
        00004C  BALR  05EF
        00004E  DC-S  0300
00118   000050  L     58F0B04C       00009C            STØP
        000054  BAL   45E0F004

***** ISAM-SAM 80 SECTIØN. *****

***** BEG 80 PARAGRAPH. *****

00122   000058  LM    9801B094       0000E4            ØPEN
        00005C  L     58F0B05C       0000AC
        000060  BALR  05EF
        000062  DC-S  0201

***** ISAM 80 PARAGRAPH. *****

00124   000064  LM    9801B094       0000E4            READ
        000068  L     58F0B060       0000B0
        00006C  BALR  05EF

SØURCE  REL     TYPE                 EFF     EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1   ADDR2     REMARKS

        00006E  DC-S  0000
        000070  DC-S  8000
00124   000072  CLC   D50020202134   001150  001264    READ
        000078  L     58D0209C       0011CC
        00007C  BC    4770D084       000084
00124   000080  BC    47F0D0C4       0000C4            GØ TØ
00125   000084  L     5840B044       000094            MØVE
        000088  MVC   D20740403058   000040  0010A8
        00008E  ØI    96F04047       000047
00126   000092  MVC   D2194005306C   000005  0010BC    MØVE
00127   000098  MVC   D21940213086   000021  0010D6    MØVE
00128   00009E  MVC   D2044000C12E   000000  00012E    MØVE
00128   0000A4  MVC   D201401FC12E   00001F  00012E    MØVE
00128   0000AA  MVC   D204403BC12E   00003B  00012E    MØVE
00129   0000B0  LA    41000048                         RELEASE
        0000B4  STH   4000C38E       00038E
        0000B8  L     58F0B06C       0000BC
        0000BC  BAL   45E0F008
00130   0000C0  BC    47F0D064       000064            GØ TØ

***** ISEND 80 PARAGRAPH. *****

00132   0000C4  BAL   45E0C062       000062            EXIT
        0000C8  DC-S  0000

***** SAM-SAM 80 SECTIØN. *****

***** SAM 80 PARAGRAPH. *****

00136   0000CA  L     58F0B06C       0000BC            RETURN
        0000CE  BAL   45E0F00C
00136   0000D2  CLI   9500201F       00114F            RETURN
        0000D6  L     58D0209C       0011CC
        0000DA  BC    4780D0E2       0000E2
        0000DE  BC    47F0D0E6       0000E6
        0000E2  BC    47F0D0EA       0000EA
00136   0000E6  BC    47F0D112       000112            GØ TØ
00137   0000EA  L     5840B040       000090            MØVE
        0000EE  MVC   D2004000C12E   000000  00012E
00138   0000F4  L     5850B044       000094            MØVE
        0000F8  MVC   D24740015000   000001  000000
00139   0000FE  LM    9801B09C       0000EC            WRITE
        000102  L     58F0B054       0000A4
        000106  BALR  05EF
        000108  DC-S  0404
        00010A  DC-S  0049
        00010C  DC-S  0000
00140   00010E  BC    47F0D0CA       0000CA            GØ TØ

***** SIEND 80 PARAGRAPH. *****

00142   000112  BAL   45E0C062       000062            EXIT
        000116  DC-S  0008
        000118  L     58F0C10C       00010C
```

```
COB1 V02.3A     COBØL-74 CØMPILATIØN     STUDY          ØBJECT PRØGRAM LISTING          15:31:09   86-05-27  PAGE 0027

                                                   PRØCEDURE DIVISIØN CØDE

SØURCE  REL     TYPE                 EFF     EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1   ADDR2     REMARKS

        00011C  BAL   45EF0008

SØURCE  REL     TYPE                 EFF     EFF
SEQ.NØ. LØCTN   INST  ØBJECT CØDE    ADDR1   ADDR2     REMARKS
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING         15:31:09   86-05-27  PAGE 0028

                 PROGRAM IDENTIFICATION, COMPILE TIME INFORMATION, LIBRARY MODULE AND PROGRAM STORAGE SUMMARY



000120  DC-C  E2E3E4C4E8F8F040 40F0F0F140C3D6D4 D7C9D3C5C440C2E8 40C3D6C2F1404040   STUDY80  001 COMPILED BY COB1
              E5C5D9E2C9D6D540 F24BF3C140404060 D9C5E57B40C14040 4040404040404040   VERSION 2.3A   -REV# A
              4040404040404040 4040404040404040 4040404040404040 4040404040404040
              40404040C24040 4040404040404040 4040404040404040 4040404040404040         B
              4040404040404040 4040404040404040 4040404040604040 D6D540F8F660F0F5                       -  ON 86-05
              60F2F740C1E340F1 F57AF3F17AF0F986 05271531093145                      -27 AT 15:31:09F.......
```

```
COB1 V02.3A     COBOL-74 COMPILATION    STUDY          OBJECT PROGRAM LISTING         15:31:09   86-05-27  PAGE 0029

                                         INTERNAL SYMBOL DICTIONARY


          SORTIEREN                           $111OPE                             BE
          $113SOR                             SORTIER                             $116CLO
          $117CLO                             $118STO                             $122OPE
          ISAM-SAM                            BEG                                 $124REA
          ISAM                                $124GOT                             $125MOV
          $126MOV                             $127MOV                             $128MOV
          $129REL                             $130GOT                             $132EXI
          ISEND                               $136RET                             SAM-SAM
          SAM                                 $136GOT                             $137MOV
          $138MOV                             $139WRI                             $140GOT
          $142EXI                             SIEND


   PROGRAM (LESS LIBRARY MODULES) OCCUPIES 00005438 BYTES.
```

Leerseite durch den Nachtrag vom August 1986

2.3.4.4 Adreßliste

Die von COB1 erzeugte Adreßliste gibt dem Anwender summarische Information über Adressen, Namen, zur Adressierung verwendete Register, Stufennummern und Datenformate für die Datendefinitionen aus dem Datenteil; außerdem werden die Namen, Adressen und Ansprechart der Prozedurnamen aus dem Prozedurteil aufgeführt. Die Auflistung der Information erfolgt im Standardfall in aufsteigender Reihenfolge der Quellprogrammfolgenummern.

Die Ausgabe der Adreßliste erfolgt standardmäßig und muß nicht vom Benutzer angefordert werden. Durch Angabe des Kommandos

```
/PARAM XREF=YES
```

vor dem Aufruf des COB1-Übersetzers wird die Ausgabe zusätzlicher Querverweisinformation in der Adreßliste angestoßen. Zu jedem Daten- bzw. Prozedurnamen werden alle Bezugnahmen auf andere Daten- bzw. Prozedurnamen aufgeführt.

Durch folgende COBRUN-Operanden kann das Listenbild der Adreßliste vom Benutzer beeinflußt werden:

COBRUN LINExx — Dieser Operand legt die maximale Zeilenzahl (2Ø ≦ nn ≦ 99) pro Seite der Übersetzungsprotokolle fest.

COBRUN MAPSRT — bewirkt, daß die Adreßliste nach Datennamen und Prozedurnamen alphanumerisch aufsteigend sortiert ausgegeben wird.

Der erste Teil der Adreßliste enthält alle Datennamen aus der FILE SECTION, WORKING-STORAGE SECTION, LINKAGE SECTION, REPORT SECTION und der SUBSCHEMA SECTION ㊸. Getrennt durch entsprechende Überschriften enthält dieser Teil der Adreßliste für jedes Kapitel des Datenteils ähnlich aufgebaute Information.

Wie aus dem folgenden Beispiel einer Adreßliste zu entnehmen ist, wird zunächst jedes Kapitel durch die entsprechende Überschriftszeile gekennzeichnet.

Im Listenteil für die FILE SECTION folgt als nächstes eine Kurzbeschreibung des FD-Namens und die Adresse des zugehörigen Dateisteuerblocks ㊹ . Die folgende Teilüberschrift ist für alle Kapitel identisch und teilt jede Zeile in folgende Bereiche ein:

SOURCE SEQ.NO. ㊺ — gibt die Quellprogrammfolgenummer der Definition an

REL ADDR ㊻ — gibt die relative Anfangsposition einer Datendefinition innerhalb einer 01-Stufe an.

{I/O | GRP} REL ADDR ㊼ — gibt die relative Anfangsadresse einer Datendefinition innerhalb des zugewiesenen Speicherbereichs an.

GEN REG ㊽ — stellt die Nummer des zur Adressierung verwendeten Registers dar. Dieses Feld bleibt leer, wenn zur Übersetzungszeit kein permanentes Register zugewiesen werden kann.

LEVEL NO. ㊾ — enthält die Stufennummer der Definition.

DATA NAME ㊿ — enthält den vom Benutzer vergebenen Datennamen.

LENGTH IN BYTES (51) — gibt die Länge des Bereiches, dem der Datenname zugeordnet wurde, in dezimaler (DEC) und in sedezimaler (HEX) Darstellung an.

OBJECT FORMAT (52) — gibt in symbolischer Form die Datenklasse an.

REFERENCED BY STATEMENTS ⑬ — Dieses Feld erscheint nur, wenn durch /PARAM XREF=YES Querverweisinformation angefordert wurde. In aufsteigender Reihenfolge sind hier alle Quellprogrammfolgenummern der Anweisungen aufgeführt, die sich auf diese Datendefinition beziehen.

Treten zu einem Datennamen mehr als 5 Querverweise auf, so wird eine Fortsetzungszeile gebildet.

Der zweite Teil der Adreßliste enthält alle Prozedur- und Kapitelnamen aus dem Prozedurteil.

Nach der Überschriftszeile PROCEDURE DIVISION teilt eine Teilüberschriftszeile die nachfolgende Information in folgende Felder ein:

SOURCE SEQ.NO. (54) — gibt die Quellprogrammfolgenummer der Definition an.

REL ADDR (55) — gibt die relative Adresse des Prozedurnamens innerhalb des zugewiesenen Speicherbereichs an.

PROCEDURE NAME (56) — enthält den vom Benutzer vergebenen Prozedur- bzw. Kapitelnamen.

KINDS OF REFERENCE (57) — gibt die Art der Bezugnahme auf einen Prozedur- bzw. Kapitelnamen an. Dabei sind folgende Angaben möglich:

NOT REFERENCED — der Prozedurname wird von keiner Anweisung angesprochen.

ENTRY POINT — der Prozedurname wird von einer GO TO-Anweisung angesprochen.

ENTRY/EXIT POINT — der Prozedurname wird mit PERFORM angesprochen.

USE REFERENCE — der Prozedurname ist Name einer Prozedurvereinbarung.

REFERENCED BY SORT — der Prozedurname ist der Name einer Ein- oder Ausgabeprozedur für eine SORT-Anweisung.

REFERENCED BY STATEMENT (58) — Dieses Feld erscheint nur, wenn durch /PARAM XREF=YES Querverweisinformation angefordert wurde. In aufsteigender Reihenfolge werden hier die Quellprogrammfolgenummern aller Anweisungen aufgeführt, die den Prozedurnamen ansprechen.

Treten für einen Prozedurnamen mehr als 6 Querverweise auf, so wird eine Fortsetzungszeile gebildet.

```
  (A)              (B)                   (C)             (D)                        (E)        (F)        (G)
 COB1 V02.3A      COBOL-74 COMPILATION  STUDY          LOCATOR MAP LISTING        15:31:09   86-05-27   PAGE 0030

                                                       DATA DIVISION (43)
                                                       FILE SECTION

                                        FILE NAME                      EINGABE (44)
                                        FILE SERIAL NO.                01
                                        ADDR LHE DTF EXP. (INPUT)      000250

                     (47)                                               (51)
  (45)      (46)     I/O     (48)            (50)                       LENGTH               (52)            (53)
 SOURCE     REL      REL     GEN   LEVEL                                IN BYTES                             REFERENCED
 SEQ.NO.    ADDR     ADDR    REG   NO.   DATA NAME                      DEC       HEX      OBJECT FORMAT     BY STATEMENTS

 00029                             FD    EINGABE                                                             00014 00088 00090 00092 00106

 00030    001058             03    01    E-SATZ                       00000072 000048                      00091

 00031    001058 000000      03    02    E-NUMMER                     00000008 000008 ZONED DECIMAL        00014

 00032    001060 000008      03    02    E-FIL1                       00000001 000001 EBCDIC-CHAR

 00033    001061 000009      03    02    E-ANZAHL                     00000005 000005 ZONED DECIMAL        00099

 00034    001066 00000E      03    02    E-REST                       00000058 00003A EBCDIC-CHAR
```

```
   (A)              (B)                 (C)              (D)                     (E)          (F)        (G)
 COB1 V02.3A    COBOL-74 COMPILATION    STUDY        LOCATOR MAP LISTING         15:31:09     86-05-27   PAGE 0031

                                                      DATA DIVISION (43)
                                                      FILE SECTION

                                     FILE NAME                      BESTAND (44)
                                     FILE SERIAL NO.                02
                                     ADDR LHE DTF EXP. (INPUT)      000700

                    (47)                                                  (51)
  (45)      (46)    I/O    (48)   (49)      (50)                        LENGTH               (52)             (53)
 SOURCE     REL     REL    GEN   LEVEL                                  IN BYTES                              REFERENCED
 SEQ.NO.    ADDR    ADDR   REG    NO.   DATA NAME                       DEC       HEX     OBJECT FORMAT       BY STATEMENTS
  00036                            FD   BESTAND                                                               00019 00088 00096 00097 00107 00116
                                                                                                              00122 00124
  00037   0010A8            03     01   B-SATZ                     00000072 000048                            00091 00093 00100
  00038   0010A8 000000         03     02   B-NUMMER                   00000008 000008 ZONED DECIMAL              00019 00096 00125
  00039   0010B0 000008     03     02   B-FIL1                     00000001 000001 EBCDIC-CHAR
  00040   0010B1 000009     03     02   B-ANZAHL                   00000005 000005 ZONED DECIMAL              00099 00099
  00041   0010B6 00000E     03     02   B-FIL2                     00000001 000001 EBCDIC-CHAR
  00042   0010B7 00000F     03     02   B-CODE                     00000004 000004 EBCDIC-CHAR
  00043   0010BB 000013     03     02   B-FIL3                     00000001 000001 EBCDIC-CHAR
  00044   0010BC 000014     03     02   B-VER                      00000026 00001A EBCDIC-CHAR                00126
  00045   0010D6 00002E     03     02   B-TIT                      00000026 00001A EBCDIC-CHAR                00127
```

```
   (A)              (B)                 (C)              (D)                     (E)          (F)        (G)
 COB1 V02.3A    COBOL-74 COMPILATION    STUDY        LOCATOR MAP LISTING         15:31:09     86-05-27   PAGE 0032

                                                      DATA DIVISION (43)
                                                      FILE SECTION

                                     FILE NAME                      AUSGABE (44)
                                     FILE SERIAL NO.                03
                                     ADDR LHE DTF EXP. (OUTPUT)     000B50

                    (47)
  (45)      (46)    I/O    (48)   (49)      (50)                        LENGTH (51)          (52)             (53)
 SOURCE     REL     REL    GEN   LEVEL                                  IN BYTES                              REFERENCED
 SEQ.NO.    ADDR    ADDR   REG    NO.   DATA NAME                       DEC       HEX     OBJECT FORMAT       BY STATEMENTS
  00047                            FD   AUSGABE                                                               00111 00117
  00048          000000            01   AUSG                       00000073 000049                            00139
  00049          000000            02   VORSCH                     00000001 000001 EBCDIC-CHAR                00137
  00050          000001            02   A-SATZ                     00000072 000048 EBCDIC-CHAR                00138
```

```
 (A)              (B)                   (C)                  (D)                      (E)        (F)        (G)
 COB1 V02.3A      COBOL-74 COMPILATION  STUDY                LOCATOR MAP LISTING      15:31:09   86-05-27   PAGE 0033

                                                             DATA DIVISION (43)
                                                             FILE SECTION

                                        SORT-FILE NAME                   SORTARB (44)
                                        FILE SERIAL NO.                  04

                     (47)
 (45)      (46)              (48)  (49)   (50)                             (51)                        (53)
 SOURCE    REL       I/O     GEN   LEVEL                             LENGTH              (52)          REFERENCED
 SEQ.NO.   ADDR      REL     REG   NO.    DATA NAME                  IN BYTES                          BY STATEMENTS
                     ADDR                                            DEC      HEX    OBJECT FORMAT
  00052                            SD     SORTARB                                                      00113 00136
  00053              000000        01     SO-SATZ                    00000072 000048                   00129 00138
  00054              000000        02     FIL1                       00000005 000005 EBCDIC-CHAR       00128
  00055              000005        02     VER                        00000026 00001A EBCDIC-CHAR       00113 00126
  00056              00001F        02     FIL2                       00000002 000002 EBCDIC-CHAR       00128
  00057              000021        02     TIT                        00000026 00001A EBCDIC-CHAR       00113 00127
  00058              00003B        02     FIL3                       00000005 000005 EBCDIC-CHAR       00128
  00059              000040        02     BES                        00000008 000008 ZONED DECIMAL     00125
```

```
 (A)              (B)                   (C)                  (D)                      (E)        (F)        (G)
 COB1 V02.3A      COBOL-74 COMPILATION  STUDY                LOCATOR MAP LISTING      15:31:09   86-05-27   PAGE 0034

                                                             DATA DIVISION
                                                             WORKING-STORAGE SECTION (43)

                     (47)
 (45)      (46)      GRP     (48)  (49)   (50)                             (51)                        (53)
 SOURCE    REL       REL     GEN   LEVEL                             LENGTH              (52)          REFERENCED
 SEQ.NO.   ADDR      ADDR    REG   NO.    DATA NAME                  IN BYTES                          BY STATEMENTS
                                                                     DEC      HEX    OBJECT FORMAT
  00062    0010F0            03    01     ANTWORT                    00000003 000003 EBCDIC-CHAR       00074 00076 00077 00078
  00063    0010F8            03    01     FELD                       00000005 000005 ZONED DECIMAL
  00001    00023E                  77     TODAYS-DATE                00000011 00000B EBCDIC-CHAR
  00002    00023E                  77     CURRENT-DATE               00000011 00000B EBCDIC-CHAR
  00003    001198                         TALLY                      00000004 000004 BINARY
  00004    00024A                  77     RETURN-CODE                00000002 000002 BINARY
  00005    001188                         SORT-FILE-SIZE             00000004 000004 BINARY
  00006    00118C                         SORT-CORE-SIZE             00000004 000004 BINARY
  00007    001190                         SORT-MODE-SIZE             00000004 000004 BINARY
  00008    001194                         SORT-RETURN                00000004 000004 BINARY
```

```
(A)                (B)                     (C)      (D)                       (E)       (F)        (G)
COB1 V02.3A        COBOL-74 COMPILATION    STUDY    LOCATOR MAP LISTING       15:31:09  86-05-27   PAGE 0035

                                                    PROCEDURE DIVISION
 (54)     (55)                                                                (58)
 SOURCE   REL      (56)                                   (57)                REFERENCED
 SEQ.NO.  ADDR     PROCEDURE NAME                         KINDS OF REFERENCES BY STATEMENTS

 00067    000004   EINLESEN SECTION                       ENTRY POINT         00066
 00068    000004       FRAGE                              NOT REFERENCED
 00073    000018       LESEN                              ENTRY POINT         00081
 00083    000004   BESTAND SECTION                        ENTRY POINT         00077 00082
 00084    000004       AUFN                               NOT REFERENCED
 00087    000004       BESTEINGABE                        NOT REFERENCED
 00089    00001C       BEST1                              ENTRY POINT         00094 00101 00104
 00095    000074       BESTPLUS                           ENTRY POINT         00093
 00098    0000A2       1                                  ENTRY POINT         00097
 00102    0000FA       FEHLER                             ENTRY POINT         00100
 00105    000118       EBEST1                             ENTRY POINT         00090
 00109    000004   SORTIEREN SECTION                      ENTRY POINT         00078 00108
 00110    000004       BE                                 NOT REFERENCED
 00112    000012       SORTIER                            NOT REFERENCED
 00120    000058   ISAM-SAM SECTION                       ENTRY POINT         00113 00113
                                                          EXIT POINT
                                                          REFERENCED BY SORT
 00121    000058       BEG                                NOT REFERENCED
 00123    000064       ISAM                               ENTRY POINT         00130
 00131    0000C4       ISEND                              ENTRY POINT         00124
 00134    0000CA   SAM-SAM SECTION                        ENTRY POINT         00113 00113
                                                          EXIT POINT
                                                          REFERENCED BY SORT
 00135    0000CA       SAM                                ENTRY POINT         00140
 00141    000112       SIEND                              ENTRY POINT         00136
```

2.3.4.5 Fehlermeldungsliste

Die von COB1 erzeugte Fehlermeldungsliste gibt dem Benutzer Aufschluß über alle während der Übersetzung von COB1 erkannten Syntax- und Semantikfehler.

Die Fehlermeldungsliste wird standardmäßig ausgegeben und muß nicht vom Benutzer angefordert werden.

Einige COBRUN-Operanden steuern die Fehlermeldungen des COB1-Übersetzers:

DIAGTEXT = {ENGLISH / GERMAN}	ermöglicht die Wahl zwischen englischen und deutschen Fehlermeldungstexten.
ERDICT	veranlaßt die Ausgabe einer Liste sämtlicher Fehlermeldungen des COB1-Übersetzers.
ERRPRn	dient zur wahlweisen Unterdrückung von Fehlermeldungen für Fehler mit SEVERITY CODE kleiner als „n".
LINEnn	legt für die Ausgabe aller Listen die maximale Zeilenzahl „nn" pro Seite fest (Standard: nn = 64).
SEQERR	veranlaßt, daß eine Fehlermeldung ausgegeben wird, sobald Quellprogrammsätze nicht in aufsteigender Reihenfolge vorgefunden werden.
WRLST	bewirkt die Ausgabe der Fehlerliste in die Systemdatei SYSLST.

Standardmäßig gibt COB1 die Fehlermeldungen auf den Drucker aus. Der Benutzer kann mit Hilfe des **PARAMETER-Kommandos** aber erreichen, daß sie in einer Datei, der Fehlerdatei, abgespeichert werden. Das hat in Dialogprozessen während der Testphase den Vorteil, daß der Benutzer nicht auf das Ausdrucken der Fehlerliste bis nach Prozeßende warten muß.

/PARAM DIAG=YES	Die Ausgabe erfolgt auf den Drucker.
/PARAM ERRFIL=YES	Die Ausgabe erfolgt in eine SAM-Datei namens "ERRFIL.COB1.progname" oder in eine Datei mit dem LINK-Namen „ERRLINK".

Die Fehlerdatei hat, bis auf fehlende Drucksteuerzeichen, den gleichen Inhalt wie die Fehlerliste. Die Datei läßt sich mit EDT oder EDOR auf die Datenstation ausgeben.

Eine ausführliche Beschreibung der Schlüsselwörter des PARAMETER-Kommandos bzw. der COBRUN-Operanden befindet sich in Abschnitt 2.4.2 bzw. 2.4.3.

Nach den Überschriftszeilen unterteilt eine Teilüberschriftszeile die nachfolgenden Fehlermeldungszeilen in folgende Bereiche:

MSG INDEX	gibt die Fehlermeldungskennzeichnung an.
SOURCE SEQ.NO.	gibt die Quellprogrammfolgenummer der Quellprogrammzeile an, in der der Fehler auftrat.
SEVERITY CODE	gibt die Fehlerklasse an (siehe Tabelle).
ERROR MESSAGES	enthält den erklärenden Text und gegebenenfalls die vom COB1-Übersetzer durchgeführte Korrektur oder einen von COB1 angenommenen Standardwert. Hier sei darauf hingewiesen, daß die wahlweise ausgegebenen deutschen Texte meistens ausführlicher sind.

Die Ausgabe der Fehlermeldungen erfolgt in aufsteigender Reihenfolge ihrer Fehlermeldungskennzeichnung.

Am Ende der Fehlermeldungsliste wird eine Abschlußinformation über Gesamtanzahl aller aufgetretenen Fehler sowie Gesamtanzahl der Fehler in den verschiedenen Fehlerklassen ausgedruckt.

```
 Ⓐ            Ⓑ                   Ⓒ            Ⓓ                        Ⓔ         Ⓕ         Ⓖ
COB1 V02.3A   COBOL-74 COMPILATION  STUDY       DIAGNOSTIC LISTING       15:31:09  86-05-27  PAGE 0036

   MSG    SOURCE   SEVERITY
  INDEX   SEQ.NO     CODE     ERROR MESSAGES

  BB220   00092       0     FUER EINE "REWRITE"- BZW. "DELETE"-ANWEISUNG WURDE WEDER DIE ANGABE "INVALID KEY" GEMACHT, NOCH
                            IST FUER DIESE DATEI EINE "USE" PROZEDUR VORHANDEN.

  TOTAL 00001 STATEMENTS IN THIS DIAGNOSTIC LISTING.

        00001 IN SEVERITY CODE 0
```

Tabelle: Fehlerklassen und ihre Bedeutung

Fehlerklasse	Beschreibung	
I	Hinweismeldung	Der Übersetzer hat Steueranweisungen oder COBOL-Sprachelemente erkannt, auf die der Anwender zwar aufmerksam gemacht werden soll, die jedoch nicht die Ausgabe einer Warnungs- oder Fehlermeldung rechtfertigen. Hinweise werden insbesondere dann gemeldet, wenn — das Quellprogramm Sprachmittel enthält oder — COBRUN-Anweisungen eingegeben wurden, die von künftigen COB1-Versionen nicht mehr unterstützt werden.
0	Warnungsmeldung	Ein möglicher Fehler wurde im Quellprogramm gemacht; trotz dieses Fehlers ist der Programmablauf möglich.
1	Fehlermeldung	Der Übersetzer hat einen Fehler entdeckt. Normalerweise macht der Übersetzer eine Korrekturannahme; ein Ablauf des Programms zu Testzwecken ist möglich.
2	Schwerwiegender Fehler	Normalerweise ist vom Übersetzer keine Korrekturannahme gemacht; die fehlerhafte Anweisung wird nicht generiert.
3	Abbruchfehler	Es ist ein so schwerwiegender Fehler aufgetreten, daß der Übersetzer nicht in der Lage ist, die Übersetzung fortzusetzen.

2.4 Ablauf

2.4.1 Übersicht über Steuerungsmöglichkeiten

Abschnitt 2.2 zeigt, daß der Benutzer zwischen verschiedenen **Eingabeformen** in den COB1-Übersetzer wählen kann. Abschnitt 2.3 stellt die Steuerungsmöglichkeiten bezüglich der **Ausgabe** des Übersetzers dar. Zusätzlich zur Ein- und Ausgabe kann der Benutzer den **Übersetzungsablauf** und damit indirekt die Form des erzeugten Objektmoduls steuern, und zwar mit dem PARAMETER-Kommando und der COBRUN-Anweisung.

Folgende Schlüsselwörter des **PARAMETER-Kommandos** steuern den Übersetzungslauf (siehe auch Abschnitt 2.4.2):

CODE=3	wirkt wie der COBRUN-Operand QUOTE1 (s. u.)

Folgende Operanden der COBRUN-Anweisung steuern den Übersetzungslauf (siehe auch Abschnitt 2.4.3):

QUOTE1	veranlaßt, daß der COB1-Übersetzer einen Apostroph (') im Quellprogramm als Anführungszeichen (") interpretiert.
SEMCHK	legt fest, daß das Quellprogramm nur syntaktisch und semantisch überprüft und kein Bindemodul erzeugt wird.
SYNCHK	bewirkt, daß das Quellprogramm ausschließlich auf syntaktische Fehler geprüft und kein Bindemodul erzeugt wird.

Die verschiedenen Möglichkeiten zur Steuerung des COB1-Übersetzers sind in folgender Übersicht zusammengestellt:

Steuerung ...	BS2000-Kommandos [2]	COBRUN-Operanden**)	Übersetzungs-anweisung [1]
... der Eingabe in COB1	/SYSFILE SYSDTA= /PARAM CODE=2*) /FILE ...,LINK=SRCLIB /FILE ...,LINK=COBLIB LINK=COBLIB1–4	SRCELEM	COPY
... des Übersetzungslaufes von COB1	/PARAM ERRFIL, ERRLST, LIST, MAP, OBJLST, SAVLST, XREF	DIAGTEXT, ERDICT, ERRLST, ERRPRn, LINEnn, MAPALL, MAPSRT, NOCOPY, NODDLIST, SEQERR, SEMCHK, SYNCHK, WRLST	
... der Modulerzeugung durch COB1	/PARAM CODE=3, DEBUG, DIAG, SYMDIC	ACTKEY, LINK, LOW#UP, MODULE, NESTPF, PARAM8, QUOTE1, RANGECHECK, SSEQ#GEN, SYMTEST, TCBENTRY, TRUNCATE	

*) siehe Abschnitt 2.4.2
**) siehe Abschnitt 2.4.3

Vor dem Aufruf des COB1-Übersetzers ist folgendes nötig:

- Eingabe des PARAM-Kommandos mit den gewünschten Operanden
- ggf. Eingabe der COBRUN-Anweisung(en) in die Eingabedatei (vor das Quellprogramm)
- ggf. Einfügen von Übersetzungsanweisungen in das Quellprogramm.

Nach diesen Vorbereitungen wird der COB1-Übersetzer schließlich mit dem EXECUTE-Kommando geladen und gestartet:

/EXEC[UTE][$]COB1

Vom Standpunkt des Betriebssystems ist der COB1-Übersetzer also ein ablauffähiges Programm (Lademodul) in einer Datei namens $TSOS.COB1, abgekürzt $COB1.

Das Beendigungsverhalten des COB1 wird davon bestimmt

- welcher Klasse die im Quellprogramm erkannten Fehler angehören und
- ob der Compiler selbst fehlerfrei abläuft.

Dieses Verhalten ist insbesondere dann von Bedeutung, wenn COB1 in einer Prozedur aufgerufen oder von einer Jobvariable überwacht wird.

Die folgende Tabelle gibt einen Überblick über die möglichen Fälle und deren Auswirkung auf den weiteren Ablauf einer Prozedur und den Inhalt der Rückkehrcode-Anzeige (siehe [22]) der Jobvariable:

Fehler	Beendigung	Dump	Rückkehrcode-Anzeige in Jobvariablen	Verhalten in Prozeduren
keine Fehler	normal	nein	0000	keine Verzweigung
Fehlerklasse I	normal	nein	0001	
Fehlerklasse 0	normal	nein	1002	
Fehlerklasse 1	normal	nein	1003	
Fehlerklasse 2	normal	nein	2004	Verzweigung zum nächsten STEP-, ABEND-, ABORT-, ENDP- oder LOGOFF-Kommando
Fehlerklasse 3	normal	nein	2005	
Compilerfehler	abnormal	ja	3006	

2.4.2 PARAMETER-Kommando

Format:
/PARAM Schlüsselwort-1=Wert-1[,Schlüsselwort-2=Wert-2]...

Die zulässigen Schlüsselwörter, Werte und deren Bedeutung sind der folgenden Tabelle zu entnehmen. Standardwerte sind unterstrichen.

Schlüsselwort	Wert	Bedeutung
CODE	1	Ohne Bedeutung; alle von COB1 erzeugten Bindemoduln sind im EBCDI-CODE codiert.
	2	Dem COB1-Übersetzer wird mitgeteilt, daß COBRUN-Anweisungen vorhanden sind.
	3	Diese Angabe ist gleichbedeutend mit: COBRUN QUOTE1.
DEBUG	YES	Dieser Operand hat nur Wirkung, falls SYMDIC=YES angegeben wurde. In diesem Fall wird durch die Angabe YES festgelegt, daß die COBOL-Verbsymbole mit Hilfe der Folgenummern des Quellprogramms (Spalte 1–6) gebildet werden sollen.
	NO	Es werden, falls SYMDIC=YES angegeben wurde, die COBOL-Verbsymbole mit Hilfe der von COB1 erzeugten Folgenummern gebildet.
DIAG[1])	YES	Nach der Übersetzung soll eine Liste der von COB1 erkannten Fehler ausgegeben werden.
	NO	Es wird keine Fehlerliste ausgegeben.
ERRFIL	YES	Die von COB1 erkannten Fehler werden in eine Fehlerdatei (SAM-Datei) mit dem Dateinamen ›ERRFIL.COB1.programmname‹ geschrieben. Diese Fehlerdatei kann nach Beendigung der Übersetzung mit EDT am Bildschirm gelesen werden. Für ›programmname‹ wird der Name aus dem PROGRAM-ID-Paragraphen des Quellprogramms verwendet. Vergibt der Benutzer keinen Programmnamen, so verwendet der Übersetzer die Prozeßfolgenummer; der Dateiname ist dann ›ERRFIL.COB1.tsn‹. Wurde bei der Übersetzung ein FILE-Kommando mit dem LINK-Namen ERRLINK gegeben, werden die Fehlermeldungen in die angegebene Benutzerdatei geschrieben.
	NO	Es wird keine Fehlerdatei erzeugt.
ERRLST	YES	Auch bei schwerwiegenden Übersetzungsfehlern (SEVERITY CODE ≧ 2) sollen Objektprogramm-, Adreß- und Querverweislisten ausgegeben werden.
	NO	Die oben angegebenen Listen werden im Falle eines Fehlers mit SEVERITY CODE ≧ 2 nicht erzeugt (siehe auch COBRUN ERRLST). In Prozeduren wird bei schwerwiegenden Übersetzungsfehlern auf das nächste STEP-Kommando bzw. auf ENDP verzweigt.
LIST[1])	YES	Der Übersetzer erzeugt eine Liste des Quellprogramms.
	NO	Es wird keine Quellprogrammliste erzeugt.

Schlüsselwort	Wert	Bedeutung
MAP[1])	YES NO	Der COB1-Übersetzer erzeugt Adreßlisten, sortiert alphabetisch nach Datennamen oder aufsteigend nach Quellprogrammfolgenummern, gesteuert durch die COBRUN-Operanden MAPALL, MAPSRT. Es wird keine Adreßliste erzeugt.
OBJLST[1])	YES NO	COB1 erzeugt eine Liste des generierten Objektprogramms. Es wird keine Objektprogrammliste erstellt.
SAVLST[2])	NO SOURCE OBJECT LOCMAP ALL	Eine Sicherstellung der von COB1 erzeugten Listen in einer katalogisierten Datei ist nicht erwünscht. Die Quellprogrammliste wird in eine SAM-Datei geschrieben, die der Benutzer nach der Übersetzung ausdrucken lassen kann. Der Name dieser katalogisierten SAM-Datei lautet ›SRCLST.COB1.programmname‹ bzw. SRCLST.COB1.tsn. (Linkname ist SRCLINK.) Eine Liste des erzeugten Objektprogramms wird in eine katalogisierte SAM-Datei mit Namen ›OBJLST.COB1.programmname‹ geschrieben bzw. OBJLST.COB1.tsn. (Linkname ist OBJLINK.) Für Adreß- und Querverweislisten wird eine katalogisierte SAM-Datei mit Namen ›LOCLST.COB1.programmname‹ erstellt. (Linkname ist LOCLINK). Alle oben angegebenen Listen werden unter den bei SOURCE, OBJECT, LOCMAP angegebenen Namen in katalogisierte SAM-Dateien geschrieben. (Linknamen wie oben). Anmerkung: Operand LIST und SAVLST sind voneinander unabhängig, d. h. LIST=YES muß nicht angegeben werden, damit SAVLST=SOURCE wirksam wird.
SYMDIC	YES NO	Es wird ein Internadreßbuch erstellt, das alle Datennamen der DATA DIVISION, sowie Paragraphen- und Kapitelnamen und die verwendeten Verben in der PROCEDURE DIVISION enthält. Entsprechende ISD-Sätze werden in den erzeugten Bindemodul aufgenommen. Damit wird die symbolische Überwachung mittels der Testhilfe IDA ermöglicht [7]. Hinweis: PARAM SYMDIC=YES ist unwirksam bei Übersetzerläufen, für die zugleich COBRUN SYMTEST=ALL angegeben wird. In diesem Fall werden keine ISD-Informationen erzeugt. Es werden keine ISD-Sätze, kein Internadreßbuch erstellt.
XREF[1])	YES NO	Die Adreßliste enthält Querverweise. Die Angabe XREF schaltet automatisch die Funktion MAP ein, sofern nicht explizit MAP=NO angegeben ist. Die Adreßliste, falls durch MAP angefordert, enthält keine Querverweise.

[1]) Alle durch diese Operanden erzeugten Listen werden von COB1 in temporäre Dateien geschrieben, die durch den System-Makro PRINT ausgegeben werden. Die Ausgabe erfolgt automatisch, angestoßen durch COB1 in einem von dem Benutzerprozeß unabhängigen Prozeß.

[2]) Eine SAVLST-Datei kann vom Benutzer durch das Systemkommando /PRINT ausgedruckt werden. Da es sich um eine katalogisierte Datei handelt, kann dieses Kommando jederzeit gegeben und wiederholt werden.
Beispiel: Ausdrucken einer Quellprogrammliste:
/PRINT SRCLST.COB1.AXEL, SPACE=E

AXEL ist der Name des Quellprogramms, wie er im PROGRAM-ID.-Paragraphen des Quellprogramms angegeben wurde.
Katalogisierte Listen sind SAM-Dateien mit Sätzen variabler Länge in Blöcken von 2048 Bytes.

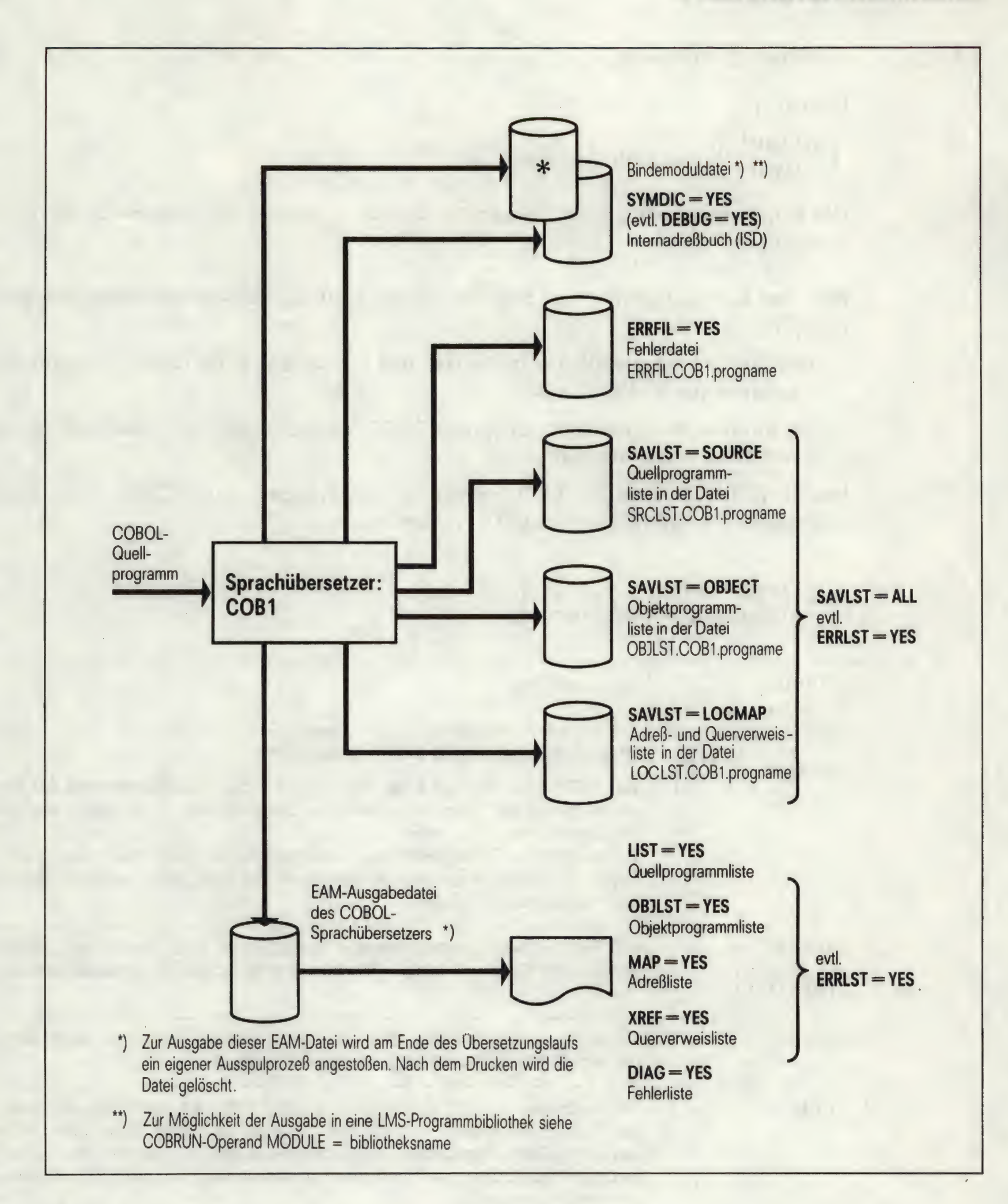


Bild 2-6
Schlüssel und Werte des PARAMETER-Kommandos

2.4.3 COBRUN-Anweisung

Format:[1])

{COBRUN | COMOPT} operand-1[,operand-2] . . .

Die Eingabe von einer oder mehreren COBRUN-Anweisungen wird durch die END-Anweisung abgeschlossen.

Mit einer Zusatzangabe in der END-Anweisung kann SYSDTA umgewiesen werden. Bei Angabe von:

- dateiname liest der COB1-Übersetzer das zu übersetzende COBOL-Programm aus der angegebenen BS2000-Datei
- bibliotheksname (elementname) liest COB1 den Quellcode aus einer LMS-Quellcode-Bibliothek unter elementname.

Wurde SYSDTA über die END-Anweisung umgewiesen, setzt COB1 zum Abschluß der Übersetzung SYSDTA auf (SYSCMD) zurück.

END [{dateiname | biblname(elementname)}]

COBRUN-Operand	Beschreibung
ACTKEY= {TTTR | TTTT}	(Betrifft nur die direkte Dateiorganisation.) Bei ACTKEY=TTTT gibt der Benutzer im Spurbezeichner des Adreßschlüssels die Nummer der „region" relativ zum Dateianfang an. Die erste region der Datei hat die Nummer 1 . . . usw. Bei ACTKEY=TTTR bleibt das niederwertigste Byte des Spurbezeichners frei (siehe [1]), so daß die region-Nummern als Vielfaches von 256 angegeben werden, beginnend mit 512.
DIAGTEXT= {ENGLISH | GERMAN}	Die Fehlermeldungstexte werden in deutscher oder englischer Sprache ausgegeben. Der Operand kann in Verbindung mit ERDICT angewendet werden.
ERDICT	Dieser Operand bewirkt, daß alle COB1-Fehlermeldungen aufgelistet werden, ohne daß eine Übersetzung durchgeführt wird.
ERRLST	Bei Auftreten einer Fehlermeldung mit SEVERITY CODE≧2 wird die Ausgabe der Objektprogramm-, Adreß- und Querverweis-Protokolle unterdrückt und nur die Fehlerliste ausgegeben. Außerdem wird die Übersetzung als fehlerhaft beendet gekennzeichnet. In Prozeduren wird auf das nächste STEP-Kommando bzw. auf ENDP verzweigt.
ERRPRn	unterdrückt Fehlermeldungen für Fehler mit einem SEVERITY CODE kleiner als n. Für n können folgende Werte angegeben werden: n=I Alle Fehler der Fehlerklassen I, 0, 1, 2, 3 werden protokolliert. n=0 Alle Fehler der Fehlerklassen 0, 1, 2, 3 werden protokolliert. n=1 Alle Fehler der Fehlerklassen 1, 2, 3 werden protokolliert. n=2 Alle Fehler der Fehlerklassen 2, 3 werden protokolliert. n=3 Alle Fehler der Fehlerklassen 3 werden protokolliert. Fehlt dieser Operand, wird n = I angenommen. Hinweis: Unterdrückte Fehlermeldungen werden mitgezählt. In der Fehlerliste wird die Meldung >PRINTING SUPPRESSED< ausgegeben.

[1]) Die Möglichkeit, anstelle von COBRUN das Schlüsselwort COMOPT zu verwenden, wird an anderen Stellen dieses Handbuches nicht mehr explizit erwähnt.

COBRUN-Operand	Beschreibung
LINEnn	Dieser Operand legt die maximale Zeilenzahl ($20 \leq nn \leq 99$) pro Seite der Übersetzungsprotokolle fest. Ein Seitenwechsel wird ausgeführt, wenn diese Zeilenzahl erreicht ist (Standard: nn = 64).
LINK	Der LINK-Name für Dateien wird aus dem Herstellerwort der ASSIGN-Klausel (siehe diese in [1]) statt aus dem Dateinamen der SELECT-Klausel entnommen.
LOW#UP	Bei Ausführung einer ACCEPT-Anweisung werden eingegebene Kleinbuchstaben in Großbuchstaben umgesetzt.
MAPALL	Dieser Operand veranlaßt die Ausgabe einer doppelten Adreßliste, die einmal aufsteigend nach Quellprogrammfolgenummern und einmal alphabetisch nach Datennamen sortiert ist. (Siehe auch MAPSRT.)
MAPSRT	Mit diesem Operanden wird die Ausgabe eines nach (vom Benutzer vergebenen) Datennamen aufsteigend sortierten Adreßbuches veranlaßt. Die ausgegebene Liste besteht aus einer Liste für die Datennamen, gefolgt von einer Liste der Kapitel- und Paragraphennamen. Bei COBRUN MAPSRT wird COBRUN MAPALL ignoriert.
MODULE= bibliotheks-name	Steuert die Ausgabe eines Objektmoduls in eine LMS-Programmbibliothek. bibliotheksname = Dateiname einer PLAM-Bibliothek.
NESTPF	Der Übersetzer generiert Befehlsfolgen für PERFORM-Anweisungen, die auf einen gemeinsamen EXIT führen. (Funktionserweiterung gegenüber dem Standard ANS74.) Achtung: Der Operand erzeugt eine umfangreichere Programmsteuerung. Dies kann zu Einbußen bei Laufzeit und zu längeren Objekten führen. Beispiel: : PERFORM A THRU C. ① : A. : PERFORM B THRU C. ② : B. : C. EXIT. ← Dieser EXIT ist für ① und ② gemeinsam.
NOCOPY	Im Quellprogramm vorhandene COPY-Bibliothekselemente werden nicht in der Quellprogrammliste abgedruckt. Die Anwendung empfiehlt sich bei häufig vorkommenden COPY-Elementen, um Papier zu sparen.
NODDLIST	Unterdrückt die Protokollierung der SUB-SCHEMA SECTION des Quellprogramms in der Quellprogrammliste. Fehlt dieser Operand, wird die Section aufgelistet und jede Zeile mit einem D gekennzeichnet.
PARAM8	Nach einer MOVE-Anweisung wird auf numerischen Feldern aus Leerzeichen die gültige Ziffer Null erzeugt.
QUOTE1	Im Quellprogramm auftretende >'< (Apostroph) werden als >"< (Anführungszeichen) interpretiert. Fehlt dieser Operand, werden die Anführungszeichen als Literalbegrenzer verwendet. Außerdem wird die Darstellung der figurativen Konstanten QUOTE gesteuert.

COBRUN-Operand	Beschreibung
RANGECHECK= {CONTINUE / TERM / <u>NO</u>}	Dieser Operand bewirkt, daß das Ablaufzeitsystem die Einhaltung von Tabellengrenzen überprüft (sowohl für Normalindizierung als auch für Spezialindizierung). Geprüft wird, ob: — Indexwerte größer als Null sind — Indexwerte nicht größer als die Anzahl von Elementen in den entsprechenden Dimensionen sind — Indexwerte nicht größer als zugehörige Werte in DEPENDING ON-Feldern sind — Werte in DEPENDING ON-Feldern innerhalb der Grenzen liegen, die in entsprechenden OCCURS-Klauseln definiert sind. Das Ablaufzeitsystem reagiert im Fehlerfall wie folgt: Meldung 9101 bzw. 9102 RANGECHECK = CONTINUE, Programmfortsetzung = TERM , Programmabbruch
SEMCHK	legt fest, daß das Quellprogramm nur syntaktisch und semantisch überprüft und kein Bindemodul erzeugt wird.
SEQERR	Werden Quellprogrammsätze in nicht aufsteigender Reihenfolge gefunden, so wird das entsprechende Satzpaar in der Fehlerliste durch eine Fehlermeldung der Fehlerklasse Ø (SEVERITY CODEØ) gekennzeichnet. Fehlt dieser Parameter, werden die Sätze mit nicht aufsteigender Satznummer nur im Protokoll des Quellprogramms (SOURCE LISTING) durch den Buchstaben ›S‹ neben der Satznummer gekennzeichnet.
SRCELEM= elementname	Unter elementname ist das zu übersetzende COBOL-Quellcodeprogramm in einer LMS-Bibliothek abgespeichert. Diese Bibliothek muß mit FILE-Kommando über den LINK-Namen SRCLIB zugewiesen sein.
SSEQ#GEN	Die Meldungen 90xx und 91xx werden ergänzt mit der von COB1 vergebenen Quellprogramm-Zeilennummer der Anweisung, bei deren Ausführung die Meldung ausgegeben wurde.
SYMTEST = {ALL / <u>MAP</u> / NO}	legt die Art der Informationen fest, die der Übersetzer für die Dialogtesthilfe AID bereitstellt. Diese Informationen werden dem Bindemodul übergeben. Sie lassen sich in zwei Teile gliedern, — eine „List for Symbol Debugging" (LSD), in der die innerhalb des Moduls definierten symbolischen Namen verzeichnet sind und — ein „External Symbol Dictionary" (ESD), das die Externbezüge des Moduls registriert. (Zum Format dieser Informationen siehe [23].) ALL: Dieser Wert muß gesetzt werden, wenn das Programm mit der Dialogtesthilfe AID symbolisch überwacht werden soll. Der Übersetzer erzeugt dann sowohl LSD- als auch ESD-Informationen, so daß beim Testen mit AID symbolische Namen aus dem Quellprogramm (wie in [25] beschrieben) verwendet werden können. Hinweise: 1. Durch COBRUN SYMTEST = ALL wird ein evtl. bestehendes PARAM SYMDIC = YES außer Kraft gesetzt, d.h. ISD-Informationen werden nicht erzeugt. 2. Bei segmentierten Programmen ist die Erzeugung von LSD-Informationen (und damit symbolisches Testen mit AID) nur dann möglich, wenn der Bindemodul mit COBRUN MODULE=bibliotheksname in eine PLAM-Bibliothek ausgegeben wird. <u>MAP</u>: Der Übersetzer erzeugt lediglich ESD-Testhilfeinformationen vom Typ Übersetzungseinheit. Dabei wird dem Objektmodul (bei segmentierten Programmen: allen Moduln) ein symbolischer Name zugeordnet, der aus den ersten 8 Zeichen des Namens im PROGRAM-ID-Paragraphen besteht. Beim Testen mit AID kann dieser Name zur Qualifikation des Quellprogrammes verwendet werden. Eine darüber hinausgehende Programmüberwachung auf symbolischer Ebene mit AID ist nicht möglich.

COBRUN-Operand	Beschreibung
	NO: Dieser Wert ist nur zu setzen, wenn das Binder-/Ladersystem Objektmoduln mit ESD-Einträgen vom Typ Übersetzungseinheit nicht verarbeiten kann. Der Übersetzer gibt dann weder LSD- noch ESD-Testhilfeinformationen an den Bindemodul weiter. Symbolisches Testen mit AID ist nicht möglich. Standardwert ist SYMTEST = MAP.
SYNCHK	Das COBOL-Quellprogramm wird nicht übersetzt, sondern nur auf syntaktische Fehler überprüft. Das Protokoll des Quellprogramms und die Fehlerliste kann man ausgeben lassen.
TCBENTRY=name	Name der Verbindungstabelle zu UTM (vgl. [10], [14]), die COB1 erzeugt. Diese Tabelle enthält Zeiger zu internen, vom Übersetzer erzeugten Arbeitsbereichen, die bei Wiederdurchlauf eines UTM-Teilprogramms von UTM erneut initialisiert werden müssen. „name" kennzeichnet den Anfang dieser Zeiger-Tabellen.
TRUNCATE= {YES | ALL | NO} (YES unterstrichen)	legt fest, wie sich die in der PICTURE-Klausel angegebene Dezimalstellenanzahl bei Datenfeldern auswirkt, für die USAGE IS COMPUTATIONAL vereinbart wurde: YES: Bei der Übertragung eines numerischen Literals in ein binäres Datenfeld wird nur die Anzahl Dezimalstellen berücksichtigt, die in der PICTURE-Klausel angegeben wurde. Überschüssige Dezimalziffern werden ggf. abgeschnitten. Beispiel: 77 EMPF PIC S999 USAGE IS COMPUTATIONAL. MOVE 1234 TO EMPF. Inhalt von EMPF: 234 ALL: Über die bei TRUNCATE = YES angegebenen Fälle hinaus wird auch bei allen MOVE-Anweisungen in binäre Felder nur die Anzahl Dezimalstellen berücksichtigt, die in der PICTURE-Klausel vereinbart wurde. Überschüssige Dezimalziffern werden ggf. abgeschnitten. NO: Bei der Übertragung eines numerischen Literals in ein binäres Datenfeld wird die Anzahl Binärstellen berücksichtigt, die für das Empfangsfeld über die PICTURE-Klausel reserviert wurde. Es werden nur dann Binärstellen des Literals abgeschnitten, wenn ihre Anzahl die tatsächliche Länge des Empfangsfeldes übersteigt. Beispiele: 1. 77 EMPF PIC S999 USAGE IS COMPUTATIONAL. MOVE 1234 TO EMPF. Inhalt von EMPF: 1234 2. 77 EMPF PIC S999 USAGE IS COMPUTATIONAL VALUE IS 65536. Inhalt von EMPF: 0 Standardwert ist TRUNCATE = YES.
WRLST	Die erzeugten Listen werden in die Systemdatei SYSLST ausgegeben (andernfalls direkt in einen eigenen SPOOL-Prozeß).

Leerseite durch den Nachtrag vom September 1985

3 Sicherstellung von Bindemoduln

Der COB1-Übersetzer erzeugt aus dem Quellprogramm ein Objektprogramm, das zwar bereits in Maschinensprache umgesetzt ist, zum Ablauf aber erst noch gebunden werden muß (siehe Kapitel 4). Die erzeugten Bindemoduln legt der Übersetzer entweder in der temporären EAM-Bindemoduldatei „*" des laufenden Prozesses oder (bei Angabe von COBRUN MODULE=bibliotheksname) in eine Programmbibliothek ab.

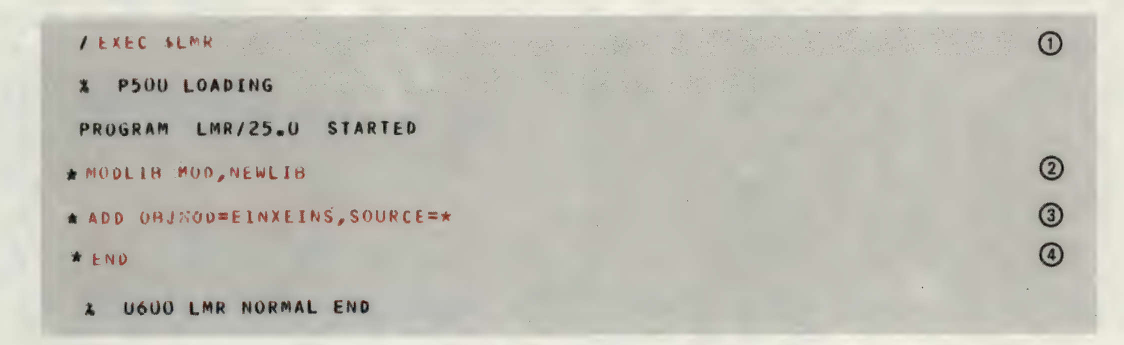
Neben dem Bibliotheksprogramm LMS und den Dienstprogrammen LMR (siehe unten) und TSOSLNK (siehe Kapitel 4) beziehen sich einige Kommandos des BS2000 [2] auf die *-Datei (ERASE *, EXEC *, LOAD *, PRINT *, PUNCH *).

Wenn in einem Prozeß ein Bindemodul, der sich in der EAM-Bindemoduldatei „*" befindet, nicht gebunden, aber über Prozeßende hinaus verfügbar bleiben, d. h. nicht bei LOGOFF mit der Datei „*" gelöscht werden soll, so bieten sich dem Benutzer drei Möglichkeiten zur Sicherstellung des Bindemoduls:

- Ausgabe auf Floppy Disk mit Hilfe des PUNCH-Kommandos,
- Ausgabe in eine katalogisierte Bindemodulbibliothek (PAM-Datei) mit Hilfe des Dienstprogramms LMR [3] oder des Bibliotheksprogramms LMS [21],
- Ausgabe in eine Programmbibliothek (Typ R) mit Hilfe des Bibliotheksprogramms LMS.

Beispiel 25: Verwendung der Routine LMR

a) In einem Dialogprozeß sind mehrere Übersetzungen gelaufen, bei denen der COB1-Übersetzer die Ergebnisse (unter anderem den Bindemodul EINXEINS) in der temporären Bindemoduldatei * speicherte.



```
/ EXEC $LMR                               ①
%  P500 LOADING
PROGRAM  LMR/25.0  STARTED
*MODLIB MOD,NEWLIB                        ②
* ADD OBJMOD=EINXEINS,SOURCE=*            ③
* END                                     ④
 %  U600 LMR NORMAL END
```

① LMR, die Routine zur Verwaltung von Bindemodulbibliotheken, wird geladen und gestartet. Die steuernden Anweisungen erwartet sie aus der Systemdatei SYSDTA, das ist hier die Datenstation.

② Als Name der Bibliothek, die bearbeitet werden soll, wird MOD vereinbart.

③ Aus der Bindemoduldatei * des Prozesses wird der **zuerst** erzeugte Modul EINXEINS in die aktuelle Bibliothek MOD übernommen.

④ Die Eingabe der LMR-Anweisungen wird beendet, ihre Ausführung beginnt. Anschließend meldet das System mit U600 das Ende des LMR-Laufes.

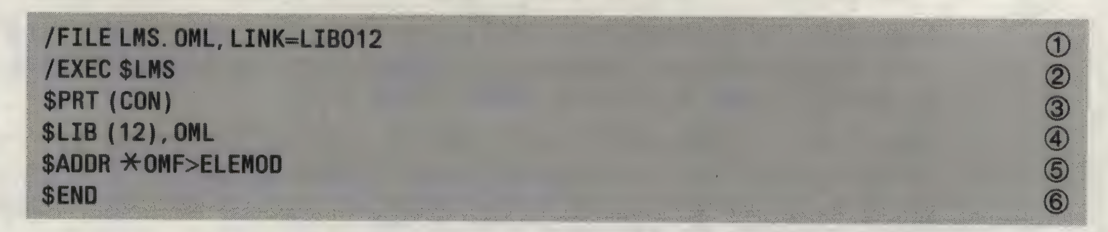
An die Stelle der temporären Datei * kann der Name einer beliebigen anderen Objektmodulbibliothek treten.

Verwendung des Bibliotheksprogramms LMS

Auch Bindemoduln können mit LMS abgespeichert werden und zwar in einer Bindemodulbibliothek oder einer Programmbibliothek. LMS bezieht Moduln aus

— der temporären EAM-Datei ✳
— einer anderen Modulbibliothek

Beispiel 26: Aufnahme eines Bindemoduls aus der ✳-Datei in eine vorhandene LMS-Bindemodulbibliothek.



```
/FILE LMS.OML,LINK=LIB012        ①
/EXEC $LMS                       ②
$PRT (CON)                       ③
$LIB (12),OML                    ④
$ADDR ✳OMF>ELEMOD                ⑤
$END                             ⑥
```

① Die Bibliothek LMS.OML wird mit dem Linknamen LIB012 verknüpft.
② Aufruf von LMS
③ LMS-Meldungen sollen nur nach SYSOUT, nicht aber nach SYSLST gehen.
④ Die Bibliothek wird als Ausgabebibliothek erklärt. NEWLIB ist nötig, falls sie neu eingerichtet wird. Durch OML wird festgelegt, daß es sich um eine Bindemodulbibliothek handeln soll.
⑤ Der **zuletzt** eingetragene Modul der ✳-Datei wird als Element ELEMOD aufgenommen.
⑥ Ende von LMS

4 Erzeugung ablauffähiger Programme

4.1 Einführung

Nach dem Übersetzen des Quellprogramms durch den COB1-Übersetzer steht das Objektprogramm dem Benutzer in der EAM-Bindemoduldatei "*" zur Verfügung, und zwar bis zum Prozeßende (LOGOFF-Bearbeitung), falls er die *-Datei nicht vorher mit dem ERASE-Kommando löscht.

Der Benutzer kann zwei Ziele verfolgen:

— Bindemoduln **sicherstellen,** dieser Fall wird in Kapitel 3 behandelt,
— Bindemoduln **binden,** d. h. ablauffähig machen.
Damit befaßt sich dieses Kapitel.

Der Bindevorgang hat die Aufgabe, aus Bindemoduln ablauffähige Programme zu erzeugen: Ein COBOL-Bindemodul kann in allen Fällen erst dann programmgemäß arbeiten, wenn er während des Ablaufs durch weitere Objektmoduln ergänzt wird. Der COB1-Übersetzer markiert dazu den Bindemodul an den Stellen durch externe Adreßkonstanten, an denen beim Programmablauf weitere Objektmoduln benötigt werden. Solche externen Adreßbeziehungen können aus mehreren Gründen aufgebaut werden:

— Der Benutzer arbeitet mit Unterprogrammen. Er legt durch eine ENTRY-Anweisung [1] oder dadurch, daß er ein COBOL-Programm als Unterprogramm verwendet, Einsprungpunkte fest, die in einem anderen Programm als externe Adressen auftreten. Handelt es sich bei dem aufrufenden Programm um ein COBOL-Programm, so wird der externe Adreßverweis durch eine CALL-Anweisung [1] veranlaßt (siehe Bild 4-1).

— Segmentiert der Benutzer ein Quellprogramm [1], so erzeugt der COB1-Übersetzer ein Wurzelsegment und ein oder mehrere Überlagerungssegmente. Das Wurzelsegment erhält entsprechende Adreßkonstanten, damit die Überlagerungssegmente beim Binden berücksichtigt und mit eingebunden werden (siehe auch Abschnitt 6.2).

— Entspricht einer COBOL-Anweisung in einem Quellprogramm kein Maschinenbefehl bzw. keine einfache Folge von Maschinenbefehlen, oder entstehen bei COBOL-Anweisungen Schnittstellen zum Betriebssystem, so erzeugt der COB1-Übersetzer die erforderlichen Maschinenbefehle nicht selbst. Er setzt an diese Stellen im Bindemodul nur externe Adreßkonstanten, die den Binder veranlassen, in der Modulbibliothek bzw. den Modulbibliotheken nach sogenannten COB1-Ablaufzeitmoduln zu suchen, die die gewünschte Funktion übernehmen. Ein solches COBOL-Quellprogramm (z. B. mit Datenbank-Sprachelementen oder mit Ein-Ausgabeoperationen) erfordert also zum Ablauf nicht nur den COB1-Übersetzer, sondern auch eine Modulbibliothek, in der COB1-Ablaufzeitmoduln stehen. Deren Gesamtheit bezeichnet man auch als **COB1-Ablaufzeitsystem.**

Die Hauptfunktion eines Binders besteht also darin, externen Adreßverweisen im Objektprogramm nachzugehen, die benötigten zusätzlichen Objektmoduln aufzusuchen, hinzuzufügen und damit das Ganze zu einer ablauffähigen Einheit zu machen.

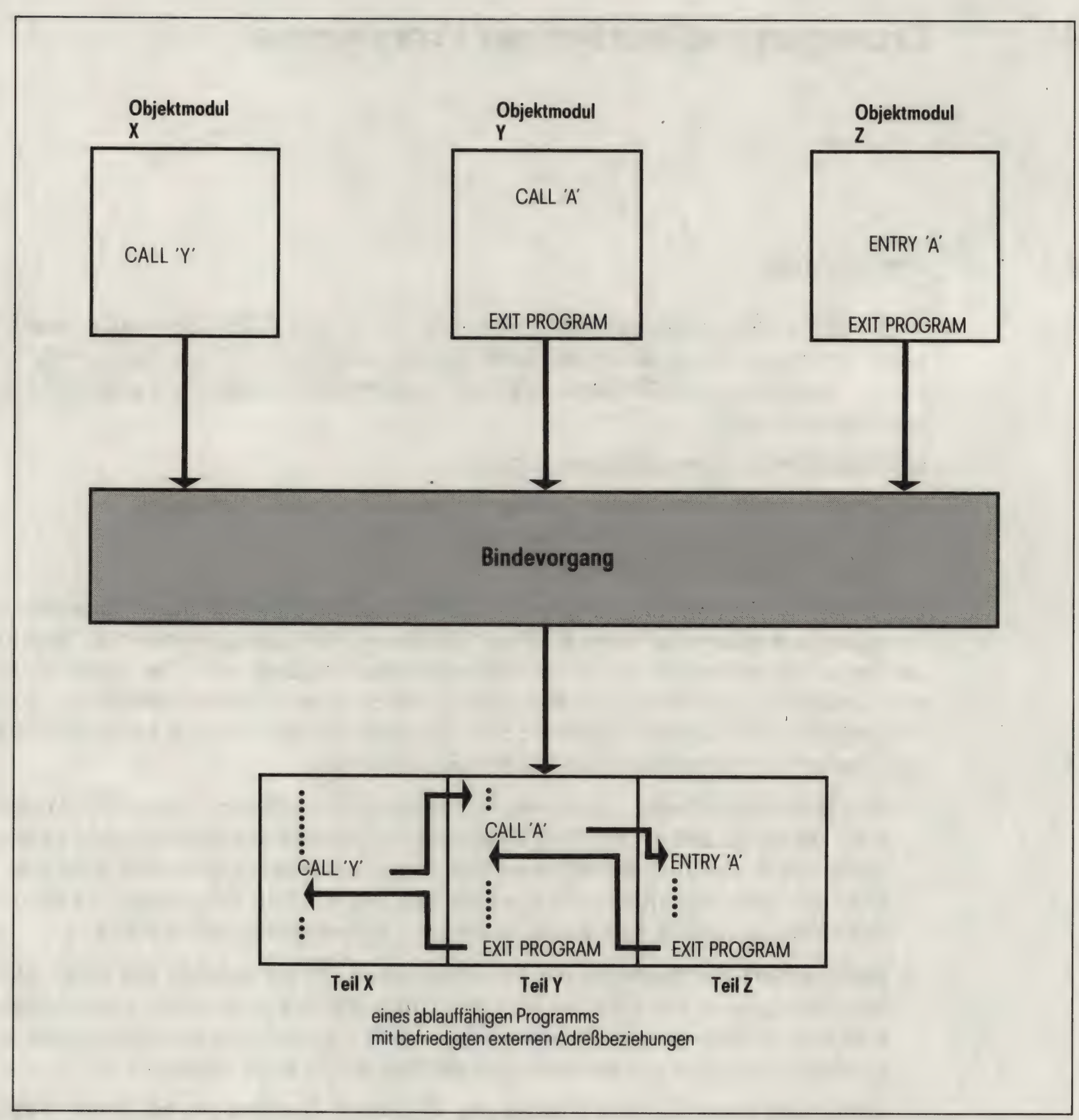


Bild 4-1:

Beispiel für die Aufgaben eines Binders

Ein Bindemodul kann vorliegen

- in temporärer Form in der EAM-Bindemoduldatei ✳
- gespeichert in einer katalogisierten Bindemodulbibliothek (erstellt durch das Dienstprogramm LMR [3] oder das Bibliotheksprogramm LMS [21])
- gespeichert in einer LMS-Programmbibliothek (erstellt durch das Bibliotheksprogramm LMS oder durch COBRUN MODULE=bibliotheksname)

Entsprechend hat der Benutzer auch beim ablauffähigen Programm die Wahl, es temporär zu machen (besonders zweckmäßig in der Testphase) oder als permanentes, ladefähiges Programm in einer Lademoduldatei zu speichern. Zwei verschiedene Binde-Routinen führen diese Aufgaben durch:

- **Der dynamische Bindelader DLL** erzeugt ein temporär ablauffähiges Programm und lädt es gleichzeitig in den Speicher (siehe Abschnitt 4.2),
- **Der statische Binder TSOSLNK** legt den erzeugten Lademodul in einer katalogisierten Datei ab, als Sonderfunktion (Anweisung MODULE) auch in der ✳-Datei (siehe Abschnitt 4.3).

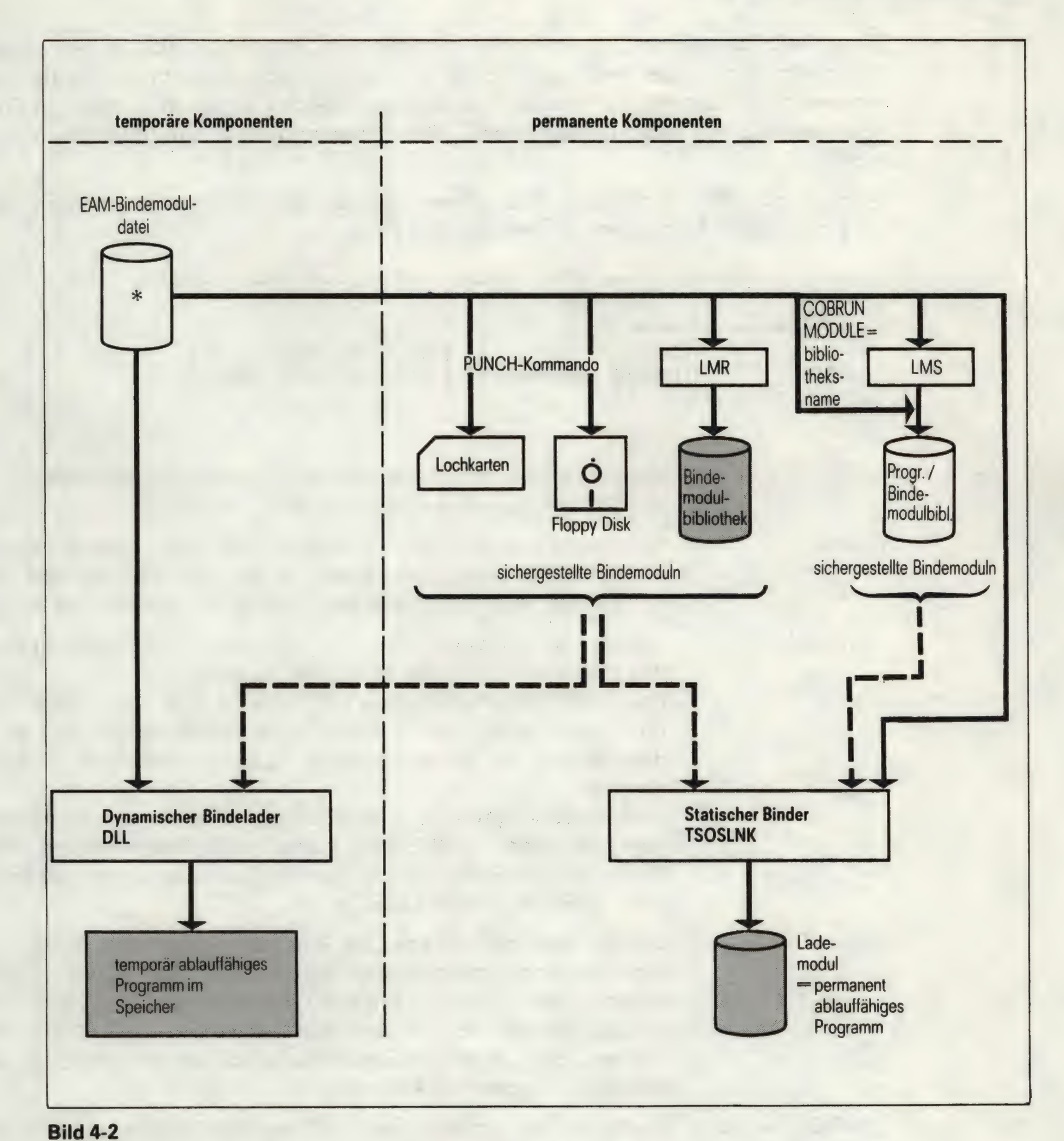


Bild 4-2

Temporär und permanent ablauffähige Programme

4

4.2 Dynamisches Binden (DLL)

Der dynamische Bindelader DLL (**D**ynamic **L**ink **L**oader) [3] erzeugt aus einem oder mehreren Objektmoduln ein ablauffähiges Programm, das nur im Speicher existiert und am Ende des Programmablaufs automatisch gelöscht wird. Wie sein Name sagt, ist der DLL für Binden **und** Laden des Programms zuständig. Er wird implizit durch BS2000-Kommandos aufgerufen, nämlich das EXECUTE- oder das LOAD-Kommando.

Das EXECUTE-Kommando [2] ruft mit folgendem Format den DLL auf und startet das von ihm erzeugte ablauffähige Programm:

Operation	Operanden
{ EXECUTE | EXEC }	{ * | (modul[,bibliothek]) } [, { IDA = {YES | NO} | SYMTEST = {ALL | NO} }] [, MONJV = jvname]

Dynamisches Binden (DLL)

Das LOAD-Kommando [24] weist den dynamischen Bindelader DLL an, ein ablauffähiges Programm zu erzeugen und in den Speicher zu laden, ohne es zu starten. Dadurch kann der Anwender vor dem Programmablauf weitere Kommandos eingeben — etwa zur Programmüberwachung mit einer Dialogtesthilfe. Das Programm kann daraufhin gestartet werden durch

- ein %RESUME-Kommando, falls mit der Dialogtesthilfe AID getestet werden soll bzw.
- ein RESUME-Kommando in allen anderen Fällen.

Operation	Operanden
LOAD	{ * | (modul[,bibliothek]) } [, { IDA = {YES | NO} | SYMTEST = {ALL | NO} }]

*	bezeichnet die temporäre Bindemoduldatei des Prozesses, in die der COB1-Übersetzer das Objektprogramm ausgibt.
modul	bezeichnet den Namen des Moduls, der zuerst geladen werden soll. modul besteht aus den ersten 8 Zeichen des Namens aus dem PROGRAM-ID-Paragraphen des zugehörigen Quellprogrammes.
bibliothek	gibt den Namen der Bindemodul- oder PLAM-Bibliothek an, in der sich das Element mit dem Namen modul befindet. Bei PLAM-Bibliotheken muß das Element vom Typ R sein. Sind mehrere Elemente gleichen Namens in der Bibliothek gespeichert, so wird das Element mit der alphabetisch höchsten Versionsbezeichnung genommen. Fehlt diese Angabe, so durchsucht der DLL die evtl. vorhandene, mit dem Operanden TASKLIB = ... des SYSFILE-Kommandos vereinbarte Bindemodulbibliothek des laufenden Prozesses, anschließend die System-Modulbibliothek TASKLIB.
IDA=YES	bewirkt, daß der dynamische Bindelader das Internadreßbuch (ISD) des Programms mit einbindet, sofern dieses Adreßbuch durch die Angabe SYMDIC = YES im PARAMETER-Kommando bei der Übersetzung erzeugt worden ist. Der Benutzer kann dann beim Testen des Programms mit IDA-Kommandos (siehe [7]) die symbolischen Adressen des Quellprogramms verwenden.
SYMTEST = ALL	ermöglicht es dem Benutzer, ohne weitere Vorkehrungen beim Testen mit AID symbolische Namen aus dem Quellprogramm zu verwenden. Voraussetzung dafür ist, daß es mit COBRUN SYMTEST = ALL übersetzt wurde.
SYMTEST = NO	ist Standardwert. Beim Testen des Programmes mit AID können nur dann symbolische Namen angegeben werden, wenn die zugehörigen Bindemoduln — bei der Übersetzung durch COBRUN SYMTEST = ALL mit LSD-Informationen versorgt wurden und — in einer PLAM-Bibliothek für das Nachladen der LSD-Informationen durch AID zur Verfügung stehen (siehe [25]).
MONJV = jvname	erklärt die Jobvariable mit dem Namen jvname zur programmüberwachenden Jobvariable (siehe [2], [22]).

Aus dem Format des EXECUTE- bzw. LOAD-Kommandos ergeben sich für den DLL zwei **Eingabequellen:**

- die EAM-Bindemoduldatei „*" des Prozesses. In den Kommandos können in diesem Fall einzelne Objektmoduln aus der *-Datei nicht ausgewählt werden.

- katalogisierte Bindemoduldatei, die mit Hilfe des Dienstprogramms LMR aufgebaut wurden.

Enthält der zu bindende Modul **externe Adreßverweise**, so hat der dynamische Bindelader DLL diese zu befriedigen. Er geht dabei folgendermaßen vor:

1. Der DLL versucht, die Adressen in der ∗-Datei bzw. in der katalogisierten Bindemoduldatei zu finden, die der Benutzer im EXECUTE- bzw. LOAD-Kommando angegeben hat.

2. Gelingt dies nicht, so sucht der DLL, ob der Benutzer mit dem SYSFILE-Kommando eine TASKLIB definiert hat:

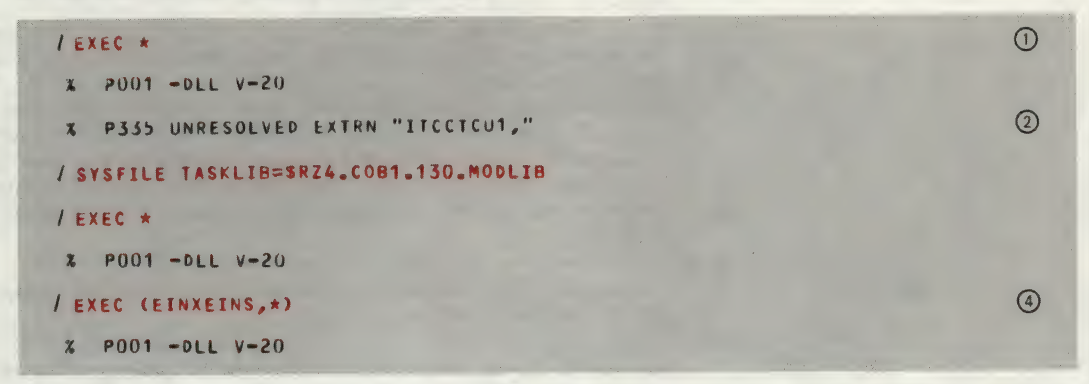
Operation	Operand
SYSFILE	TASKLIB = { dateiname / (NO) }

   Mit „dateiname" wird der Name einer katalogisierten Bindemoduldatei vereinbart, mit (NO) diese Zuordnung rückgängig gemacht.

   Existiert eine solche TASKLIB für den Prozeß, so versucht der DLL die bis dahin unbefriedigten Adreßverknüpfungen mit Hilfe der in ihr enthaltenen Bindemoduln herzustellen.

3. Zuletzt sucht der DLL die Adreßverweise mit der System-Bindemodulbibliothek TSOS.TASKLIB zu befriedigen.

Beispiel 27: dynamisches Binden (DLL)



```
/ EXEC *                                              ①
 %  P001 -DLL V-20
 %  P355 UNRESOLVED EXTRN "ITCCTCU1,"                 ②
/ SYSFILE TASKLIB=$RZ4.COB1.130.MODLIB
/ EXEC *
 %  P001 -DLL V-20
/ EXEC (EINXEINS,*)                                   ④
 %  P001 -DLL V-20
```

① Der erste Bindemodul aus der temporären Bindemoduldatei ∗ des Prozesses soll gebunden werden und ablaufen.

② Der DLL meldet, daß externe Adreßverweise (EXTRN) zu COB1-Ablaufmoduln (ITC...) nicht befriedigt werden können, d.h. sie sind in der Objektmodulbibliothek namens TASKLIB nicht vorhanden.

③ In diesem Rechenzentrum sind die Moduln des COB1-Ablaufzeitsystems in der Datei $RZ4.COB1.130.MODLIB gespeichert. Sobald diese zur TASKLIB erklärt worden ist, läuft das Kommando EXEC ∗ erfolgreich.

④ Der Bindemodul EINXEINS aus der temporären Bindemoduldatei ∗ des Prozesses soll ablaufen.

Beispiel 27a: Dynamisches Binden und Starten eines in einer Bibliothek abgelegten Bindemoduls

```
/SYSFILE TASKLIB=$COB1MODLIB                          ①
/EXEC (ELEMOD,LMS.OML)                                ②
```

① Zuweisung der TASKLIB auf die COB1-Modulbibliothek
② Der unter dem Elementnamen ELEMOD abgelegte Bindemodul soll ablaufen.

4.3 Statisches Binden (TSOSLNK)

Der statische Binder TSOSLNK (TSOS Linkage Editor) [3] erzeugt aus einem oder mehreren Objektmoduln ein ablauffähiges Programm, den sogenannten Lademodul, und speichert ihn in einer Datei ab. Der Inhalt dieser Datei ist dann jederzeit lade- und ausführbar.

Der Binder TSOSLNK gehört zu den Dienstprogrammen und wird mit dem EXECUTE-Kommando geladen und gestartet:

/EXEC $TSOSLNK

Danach erwartet der Binder Anweisungen aus der Systemdatei SYSDTA, die seinen Lauf steuern müssen.

Im folgenden werden die wichtigsten Anweisungen genannt, weitere Binder-Anweisungen sind in Druckschrift [3] bzw. [24] beschrieben.

a) Für **implizites Binden,** bei dem der Benutzer die Struktur des Lademoduls, das heißt seine Segmentierung, dem System überläßt („Normalfall" im BS2000), sind folgende Anweisungen wichtig:

Anweisung	Beschreibung
{ PROGRAM / PROG }	legt fest, wie das Ergebnis des Binderlaufes aussehen soll. Die dazu nötigen Operanden werden in [24] näher erläutert. Anmerkungen: — CONTROL = N bewirkt, daß der Binder keinen Steuermodul zum Nachladen von Programmteilen einbindet. Seine Angabe ist gleichbedeutend mit der Binderanweisung NOCTL. — SYMTEST = ALL oder SYMTEST = MAP ermöglichen es dem Benutzer, beim Testen mit der Dialogtesthilfe AID die symbolischen Namen aus dem Quellprogramm zu verwenden: Voraussetzung dafür ist, daß COB1 beim Übersetzen des Quellprogrammes durch COBRUN SYMTEST=ALL veranlaßt wurde, SD-Informationen zu erzeugen. SYMTEST = ALL weist den Binder an, diese Informationen an den Lademodul weiterzugeben, während SYMTEST = MAP bewirkt, daß im Testfall SD-Informationen aus dem Bindemodul nachgeladen werden können (siehe dazu [25]).
INCLUDE	legt einzelne Objektmoduln bzw. Objektmodulbibliotheken fest, aus denen der Binder den Lademodul aufbauen soll.
RESOLVE und EXCLUDE	steuern, wo der Binder automatisch Objektmoduln suchen soll, wenn er externe Adreßverweise findet, die nicht durch INCLUDE-Anweisungen befriedigt werden (=Autolink-Funktion des Binders). RESOLVE gibt Objektmodulbibliotheken für das Autolink-Verfahren an, EXCLUDE schließt Bibliotheken davon aus. Anmerkung: Eine EXCLUDE-Anweisung für die TASKLIB, die Bindemodulbibliothek des Systems, ist identisch mit der Binderanweisung NCAL.
END	markiert das Ende der Eingabe von Binderanweisungen.

b) Beim **expliziten Binden** legt der Benutzer mit den folgenden Anweisungen, die er zusätzlich zu den unter a) genannten geben kann, weitere Eigenschaften des Lademoduls fest:

Anweisung	Beschreibung
OVERLAY	beschreibt den Aufbau eines segmentierten Programms.
TRAITS	vereinbart für einen Programmteil, daß er a) beim Laden auf Seitengrenze ausgerichtet werden soll, b) während des Programmlaufs nur gelesen werden darf („read only“).

Explizites Binden ist im BS2000 nur in seltenen Fällen notwendig, da durch den Seitenwechsel-Mechanismus das Betriebssystem automatisch „nachlädt“, d.h. es befinden sich nur die gerade benutzten 4K-Seiten des Programms im Hauptspeicher. Sobald weitere Seiten des Programms benötigt werden, lädt das System sie aus dem Seitenwechselbereich der Magnetplattenspeicher in den Hauptspeicher. Der Benutzer merkt im allgemeinen nichts von diesem Vorgang.

Findet der Binder in einem Objektmodul **externe Adreßverweise**, die nicht durch die Moduln befriedigt werden können, die der Benutzer in INCLUDE-Anweisungen genannt hat, so geht der Binder nach folgendem **Autolink-Mechanismus** vor:

- Als erstes prüft der Binder, ob der Benutzer in einer RESOLVE-Anweisung dem externen Adreßverzeichnis explizit eine Bindemoduldatei zugeordnet hat, in der ein passender Modul zu finden ist.
- Bleibt dabei die Suche nach einer Einsprungadresse erfolglos, geht der Binder sämtliche Bindemoduldateien durch, die der Benutzer in RESOLVE-Anweisungen genannt hat. Dies kann der Benutzer mit entsprechenden EXCLUDE-Anweisungen unterdrücken.
- Ist es dem Binder immer noch nicht gelungen, den externen Adreßverweis zu befriedigen, durchsucht er die Bindemoduldatei TASKLIB, falls der Benutzer es nicht mit einer entsprechenden EXCLUDE-Anweisung bzw. mit einer NCAL-Anweisung verhindert. Die Datei TASKLIB wird zuerst unter der Benutzerkennung des Prozesses gesucht. Ist dort keine Datei dieses Namens vorhanden, wird die Bindemoduldatei des Systems, die Datei $TSOS.TASKLIB, genommen.

Alle gefundenen Bindemoduln bindet der TSOSLNK zum Lademodul. Bleibt seine Suche erfolglos, gibt er Fehlermeldungen aus.

Es ist nicht erlaubt, COBOL-Programme als Klasse 1-Programme zu binden.

Beispiel 28: statisches Binden (TSOSLNK)

```
/SYSFILE SYSDTA=(PRIMARY)                                   ①
/EXEC $TSOSLNK                                              ②
% P500 LOADING
VERS. 0012 OF BS2000  LINK EDITOR READY
*PROG XYZ                                                   ③
*INCLUDE EINXEINS,*                                         ④
*RESOLVE ,$RZ4.COB1.130.MODLIB                              ⑤
*END
PROGRAM BOUND                                               ⑥
PROGRAM FILE WRITTEN: XYZ
NUMBER PAM PAGES USED:      5
/EXEC XYZ                                                   ⑦
```

① Der Binder TSOSLNK soll seine Anweisungen von der Eingabequelle lesen, von der auch die Kommandos kommen. SYSDTA wird deshalb mit SYSCMD zusammengeschaltet.

② Das Dienstprogramm TSOSLNK wird geladen und gestartet.

③ XYZ vereinbart den Namen des gebundenen Programms, gleichzeitig aber auch den Namen der Ausgabedatei, in die das gebundene Programm geschrieben werden soll.

④ Der Bindemodul EINXEINS aus der Datei * soll gebunden werden.

⑤ In der Bibliothek $RZ4.COB1.130.MODLIB sucht der Binder Einsprungadressen.

⑥ Die Binder-Anweisungen werden beendet. Der Binder meldet die Erstellung der Ausgabedatei.

⑦ Das Programm XYZ wird gestartet.

Beispiel 28a: Statisches Binden eines in einer Bibliothek abgelegten Bindemoduls und Starten des Programms

```
/SYSFILE SYSDTA=(SYSCMD)                                    ①
/EXEC $TSOSLNK                                              ②
*PROG COBPROG,FILENAM=L.PROG                                ③
*INCLUDE ELEMOD,LMS.OML                                     ④
*RESOLVE ,$COB1MODLIB                                       ⑤
*END
/EXEC L.PROG                                                ⑥
```

① Anweisungen für TSOSLNK sollen von der gleichen Eingabequelle kommen wie Kommandos.
② Aufruf des Binders.
③ Das Programm wird COBPROG genannt und soll auf der Datei L.PROG stehen.
④ Der Modul ELEMOD aus der Bibliothek LMS.OML wird gebunden.
⑤ In der COB1-Modulbibliothek COB1MODLIB sucht der Binder Einsprungadressen.
⑥ Starten des Programms.

5 Handhabung ablauffähiger Programme

5.1 Zuweisung von Betriebsmitteln

5.1.1 Übersicht

Eine der wesentlichen Aufgaben von COBOL-Programmen in der kommerziellen Datenverarbeitung ist die Ein-Ausgabe von Daten bzw. der Zugriff zu Dateien. Dazu ist eine entsprechende Programmierung erforderlich (siehe Kapitel 6). Der Benutzer muß vor dem Aufruf des ablauffähigen Programms die benötigten Betriebsmittel zuweisen. Abschnitt 5.1 behandelt die wichtigsten Typen der Betriebsmittelzuweisung:

Das Programm soll

- Daten aus den Systemdateien SYSDTA oder SYSIPT einlesen oder in die Systemdateien SYSLST, SYSLSTnn, SYSOUT oder SYSOPT ausgeben (siehe Abschnitt 5.1.2);
- Daten aus Dateien lesen oder in Dateien schreiben (siehe Abschnitt 5.1.3).

5.1.2 Bearbeitung von Systemdateien

Systemdateien sind normierte Dateien oder Geräte. Sie verwendet der Benutzer, um kleine Datenmengen ein- oder auszugeben. Im Quellprogramm verwendet er dazu (ohne Dateierklärung und ohne OPEN oder CLOSE) die COBOL-Anweisungen ACCEPT, DISPLAY, STOP-Literal oder die Testhilfe-Anweisungen EXHIBIT und TRACE:

- ACCEPT kann auf SYSDTA, SYSIPT oder den Bedienungsplatz zugreifen.
- DISPLAY kann auf SYSLST, SYSLSTnn, SYSOPT, SYSOUT oder am Bedienungsplatz ausgeben.
- Das Literal der STOP-Literal-Anweisung wird am Bedienungsplatz ausgegeben.
- Die Testhilfe-Anweisungen EXHIBIT und TRACE veranlassen bzw. steuern Ausgaben nach SYSLST.

Die folgenden beiden Tabellen zeigen, welche Systemdateien bei der Eingabe bzw. bei der Ausgabe möglich sind.

Tabelle: Systemdateien und COBOL-Anweisungen für die Eingabe

Systemdatei	COBOL-Anweisung	Bedeutung
SYSDTA	ACCEPT... FROM { SYSRDR[1]) / SYSIN / TERMINAL / merkname[2]) }	Eingabe von Daten vom Datensichtgerät, aus der SYSDTA bzw. SYSIPT zugewiesenen Datei oder vom Lochkartenleser
SYSIPT	ACCEPT... FROM { SYSIPT / merkname[2]) }	
Bedienungsplatz	ACCEPT... FROM CONSOLE[3])	Eingabe am Bedienungsplatz

Systemdateien

Tabelle: Systemdateien und COBOL-Anweisungen für die Ausgabe

Systemdatei	COBOL-Anweisung	Bedeutung
SYSLST	EXHIBIT TRACE[4]) DISPLAY...UPON { SYSLST / merkname[2]) }	Ausgabe auf Drucker oder in SAM-Datei
SYSLSTnn[5])	DISPLAY . . . UPON merkname[2])	Ausgabe auf Drucker oder in SAM-Datei
SYSOUT	DISPLAY...UPON { SYSOUT / TERMINAL / merkname[2]) }	Ausgabe auf — Datensichtgerät (im Dialogbetrieb) — Drucker (im Stapelbetrieb)
SYSOPT	DISPLAY. . . UPON { SYSOPT / SYSPUNCH / SYSPCH / merkname[2]) }	Ausgabe von Daten im Lochkartenformat; Ausgabe in SAM-Datei, auf Floppy Disk oder auf Lochkartenstanzer
Bedienungsplatz	STOP literal DISPLAY. . . UPON CONSOLE[3])	Ausgabe am Bedienungsplatz

[1]) SYSRDR wird aus Kompatibilitätsgründen zum BS1000 unterstützt.
[2]) Wie im SPECIAL-NAMES-Paragraphen festgelegt.
[3]) CONSOLE sollte möglichst nicht verwendet werden, da im „Closed Shop"-Betrieb des BS2000 der Operateur meist nicht über Dateneingabe und Datenausgabe Bescheid weiß.
[4]) Betrifft die durch READY TRACE eingeschalteten und durch RESET TRACE ausgeschalteten Testüberwachungsausgaben.
[5]) nn muß eine Zahl zwischen 01 und 99 sein. SYLST01,...,SYSLST99 sind logische Ausgabedateien des Betriebssystems ab Version 8.0.

Beispiele zur Programmierung: Beispiel 1 und Beispiel 38 b in Abschnitt 6.4.

Zuordnung der Systemdateien zu Geräten

Standardmäßig sind im BS2000 die Systemdateien SYSDTA, im Stapelbetrieb SYSIPT, SYSLST und SYSOUT bei Prozeßbeginn einem bestimmten Eingabe- bzw. Ausgabe-Gerät zugeordnet. Man bezeichnet dies als Primärzuweisung (siehe Tabelle).

Diese Systemdateien können mit dem SYSFILE-Kommando vom Benutzer umgewiesen bzw. neu zugewiesen werden (ausführliche Beschreibung im Manual „Kommandosprache", [2]). Das dritte Format in der nachstehenden Tabelle legt fest, daß nach Prozeßende SYSOPT auf eine Floppy Disk geschrieben wird. Format:

```
                        { dateiname   }
                        { (SYSCMD)    }
/SYSFILE { SYSDTA } =   { (PRIMARY)   }
         { SYSIPT }     { (mn)        }
                        { (CARD)      }
---------------------------------------------
         { SYSLST    }     { dateiname             }
/SYSFILE { SYSLSTnn  }  =  { (dateiname,EXTEND)    }
         { SYSOPT    }     { *DUMMY                }
                           { (PRIMARY)             }
/SYSFILE SYSLSTnn = *SYSLSTpp
---------------------------------------------
/SYSFILE DEVICE=DISKETTE,FILE=SYSOPT
```

dateiname — Name einer katalogisierten Datei, der SYSDTA, SYSIPT, SYSLST, SYSLSTnn (01 ≤ nn ≤ 99) oder SYSOPT zugewiesen werden soll.

EXTEND — Eine bereits vorhandene Datei soll erweitert werden.

*DUMMY — Ausgabe in eine Pseudodatei, d. h. keine Ausgabe auf externe Speicher.

(SYSCMD) — schaltet die links vom Gleichheitszeichen stehende Systemdatei mit der Systemdatei SYSCMD zusammen; das ist die Datei, von der Kommandos gelesen werden (vgl. „Kommandosprache", [2]).

(PRIMARY) — legt die betreffende Systemdatei auf ihre Primärzuweisung zurück.

(mn) ordnet SYSDTA einen Kartenleser bzw. ein Disketten-Gerät (ab BS2000 V7.1) mit einem 2 Bytes langen mnemotechnischen Namen zu.

(CARD) weist den Kartenleser zu.

*SYSLSTpp logische Ausgabedatei des Betriebssystems ab Version 8.0, der SYSLSTnn zugeordnet werden soll. Folgende Bedingungen müssen dabei erfüllt sein:
1. nn und pp müssen zweistellige Zahlen zwischen 01 und 99 sein.
2. nn ≠ pp
Eine ausführliche Beschreibung dieses Formats findet sich in [2].

Anmerkung: SYSIPT hat im Dialogbetrieb keine Primärzuweisung.

Tabelle: Systemdateien, Primärzuweisungen, Umweisungen

System-datei	Primärzuweisung		Umweisung auf	mit dem Kommando
	im Dialogbetrieb	im Stapelbetrieb		
SYSDTA	Datenstation	SPOOLIN-Datei oder ENTER-Datei	katalogisierte Plattendatei (SAM oder ISAM)	/SYSFILE SYSDTA=dateiname = bibl (elem)
			Kartenleser	... =(CARD) oder ... =(mn)
			Floppy Disk	... =(mn)
SYSIPT	–	SPOOLIN-Datei oder ENTER-Datei	katalogisierte Plattendatei (SAM oder ISAM)	/SYSFILE SYSIPT=dateiname
			Kartenleser	... =(CARD) oder ... =(mn)
SYSOUT	Datenstation	temporäre SPOOLOUT-Datei (EAM-Datei), die bei Prozeßende auf dem Drucker ausgegeben und anschließend gelöscht wird	katalogisierte Datei (Band oder Platte)	/SYSFILE SYSOUT = dateiname (ab BS2000 V7.1) nur im Stapelbetrieb
SYSLST SYSLSTnn [1])	temporäre SPOOLOUT-Dateien (EAM-Dateien), die bei Prozeßende auf dem **Drucker** ausgegeben und anschließend gelöscht werden.		katalogisierte Plattendatei (SAM)	/SYSFILE SYSLST[nn] = dateiname ... = (dateiname, EXTEND)
			Pseudodatei (*DUMMY)	/SYSFILE SYSLST[nn] = *DUMMY
SYSOPT	temporäre SPOOLOUT-Datei (EAM-Datei), die bei Prozeßende auf dem **Kartenstanzer** ausgegeben und anschließend gelöscht wird		katalogisierte Plattendatei (SAM)	/SYSFILE SYSOPT=dateiname; ... =(dateiname, EXTEND)
			Pseudodatei (*DUMMY)	/SYSFILE SYSOPT= *DUMMY
			Floppy Disk	/SYSFILE FILE=SYSOPT, DEVICE=DISKETTE

[1]) nn muß eine Zahl zwischen 01 und 99 sein. SYLST01,...,SYSLST99 sind logische Ausgabedateien des Betriebssystems ab Version 8.0.

Ändern des Ausgabeformats

Überdies kann mit dem SYSFILE-Kommando zur Ausgabe über SYSLST (Dateien im Drukkerformat) bzw. über SYSOPT (Dateien im Lochkartenformat) das standardmäßige Format geändert werden (näheres siehe Manual „Kommandosprache", [2]):

Operation	Operanden
SYSFILE	[FILE= {SYSLST | SYSOPT}] [,PRINTER= {136 | 160}] [,HREC=m] [,FORM=code][,LOOP=vfb] [,COPIES= {anzahl1 | (anzahl1, anzahl2)}] [,CHARS=(z1[,z2][,z3][,z4])] [,CONTROL= NO|PHYS] [,IMAGE= xxxx] [,DIA = zz] [,SHIFT = spalten]

5.1.3 Bearbeitung von Dateien

Ein COBOL-Programm kann Dateien verschiedener Organisationsformen mit den Zugriffsmethoden SAM, ISAM oder PAM bearbeiten lassen.

Organisationsform im COBOL-Programm (ORGANIZATION-Klausel)	Zugriffsmethode des Datenverwaltungs-system im BS2000
sequentiell (SEQUENTIAL) Sätze werden in der Reihenfolge gelesen, in der sie in der Datei stehen; sie werden entsprechend der Reihenfolge der WRITE-Anweisungen in die Datei geschrieben.	SAM (Platte oder Band)
relativ (RELATIVE) Sätze erhalten eine relative Satznummer, mit deren Hilfe zugegriffen wird.	PAM (Platte)
indiziert (INDEXED) Jeder Satz enthält ein Schlüsselfeld, aus dem sich das Datenverwaltungs-system einen Index aufbaut und mit dessen Hilfe zugegriffen wird.	ISAM (Platte)

Als Informationen über die Datei, die verarbeitet werden soll, enthält das COBOL-Programm:

- die Dateierklärung (FD),
- Angaben in der SELECT-Klausel,
- Angaben in den APPLY-Klauseln (nur bei indizierten Dateien).

In Kapitel 6 wird ausführlich gezeigt, auf welche Dateitypen ein COBOL-Programm zugreifen kann und wie es dazu programmiert werden muß.

Im allgemeinen gibt der Benutzer vor dem Programmaufruf je ein FILE-Kommando [2] für die zu verarbeitenden Dateien, in dem er Dateinamen, evtl. auch Betriebsmittel und Dateieigenschaften festlegt.

Wird kein FILE-Kommando gegeben, nimmt das System den COBOL-Dateinamen als Namen der katalogisierten Datei.

Das Format des FILE-Kommandos mit genauer Beschreibung der Operanden ist in [2] nachzulesen.

Hinweise zu einigen Operanden:

„dateiname" ist ein vom COBOL-Programm unabhängiger, frei wählbarer Name. Er ist im Katalog eingetragen bzw. wird durch das Kommando neu in ihn aufgenommen.

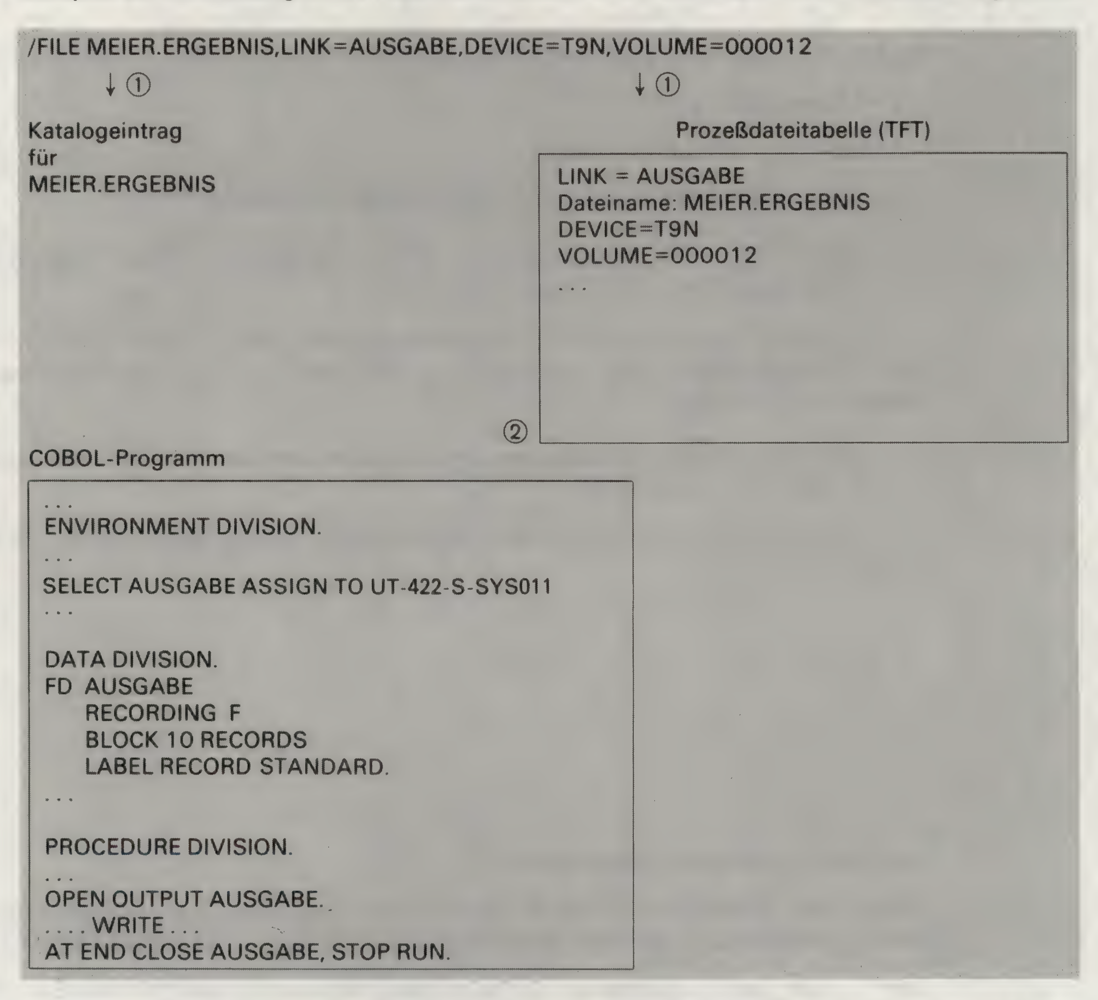
„link" ist der Dateikettungsname (LINK-Name). Mit seiner Hilfe stellt das System während des Programmlaufs die Beziehung zwischen den übrigen Angaben im FILE-Kommando und den Angaben zur Datei her, die sich im Programm befinden. Dazu muß „link" mit den ersten 8 Zeichen eines COBOL-Dateinamens in der SELECT-Klausel übereinstimmen. Ausnahmen: Übersetzt man mit dem COBRUN-Operanden LINK, muß „link" mit NAME1 (maximal 8 Zeichen) im Herstellerwort der ASSIGN-Klausel übereinstimmen (siehe BS2000 COB1 Beschreibung [1]). Folgende Dateikettungsnamen (=COBOL-Dateinamen) sind für das Dienstprogramm SORT reserviert und dürfen innerhalb eines Programms mit SORT für andere Dateien nicht verwendet werden:

MERGExx (xx = 01 ,..., 99)
SORTIN,SORTINxx (xx = 01 ,..., 99)
SORTOUT
SORTWK,SORTWKx (x = 1 ,..., 9), SORTWKxx (xx = 01 ,..., 99)
SORTCKPT

Alle übrigen Operanden des FILE-Kommandos beschreiben die Betriebsmittel der Datei und die Dateieigenschaften.

Die Angaben des FILE-Kommandos speichert das Betriebssystem unter dem Dateikettungsnamen in der Dateitabelle des Prozesses (TFT, task file table). Dort verbleiben sie bis Prozeßende, bis zu einem RELEASE-Kommando oder einem neuen FILE-Kommando, das sich auf diesen Dateikettungsnamen bezieht. Sobald eine Datei eröffnet wird (OPEN im COBOL-Programm), prüft das System nach, ob der COBOL-Dateiname als Kettungsname in der Prozeßdateitabelle (TFT) vorkommt. Bei Übereinstimmung berücksichtigt das System die Vereinbarung des FILE-Kommandos: Sie überschreiben für diesen Programmlauf die Angaben im Programm, so daß zum Beispiel zu einem im Kommando festgelegten Gerät zugegriffen wird, auch wenn ursprünglich durch ASSIGN ein anderes vereinbart worden ist.

Beispiel 29: Verkettung eines FILE-Kommandos mit Dateiangaben im COBOL-Programm



① Bei der Bearbeitung des FILE-Kommandos wird unter anderem der Eintrag AUSGABE in der Prozeßdateitabelle aufgebaut.

② Bei Dateieröffnung (OPEN) überschreiben die Angaben der Prozeßdateitabelle, die zum Eintrag AUSGABE gehören, die Angaben im COBOL-Programm.

Der LINK-Mechanismus erweist sich als vorteilhaft, weil man bei jedem Programmlauf andere Dateiattribute festlegen kann, ohne das gespeicherte Programm zu ändern, d.h., ohne neu übersetzen zu müssen.

Beispiel 30: Betriebsmittelzuweisung bei Bandverarbeitung (vgl. Beispiel 29)

```
/FILE MEIER.ERGEBNIS,LINK=AUSGABE,DEVICE=T9N,VOLUME=000012     ①
/CAT MEIER.ERGEBNIS,STATE=UPDATE,RDPASS=C'BEN'                 ②
/PASSWORD C'BEN'                                               ③
/EXEC (VERARB,BIBLIO)                                          ④
```

① Dateiname, Gerät (DEVICE) und Datenträger (VOLUME) werden mit dem Dateikettungsnamen AUSGABE verknüpft, d. h. es wird ein entsprechender Eintrag in die Dateitabelle des Prozesses (TFT) gemacht. Außerdem wird die Datei MEIER.ERGEBNIS in den Katalog aufgenommen.

② Unabhängig von der Bearbeitung des FILE-Kommandos wird für die Datei MEIER. ERGEBNIS ein Lese-Kennwort C'BEN' in den Katalog eingetragen, d. h. nur der Benutzer kann die Datei lesen oder in sie schreiben, der dieses Kennwort dem System bekannt gibt.

③ Das Kennwort C'BEN' wird dem System mitgeteilt.

④ Der Objektmodul VERARB aus der Bibliothek BIBLIO wird gebunden und gestartet. Er besitzt für die Ausgabe den COBOL-Dateinamen AUSGABE und verarbeitet daher die Banddatei MEIER.ERGEBNIS auf dem Datenträger 000012.

Um mehrere Dateien auf mehreren Bändern zu verarbeiten, verwendet der Benutzer die MULTIPLE FILE TAPE CONTAINS-Klausel (siehe COB1 Sprachbeschreibung, [1]). Benötigte Operanden für die entsprechenden FILE-Kommandos sind im Manual „Kommandosprache“, [2] beschrieben

Beispiel 31: Zugriff auf die dritte Datei auf einer Folge von zwei Bändern

```
/FILE DATEI.A,LINK=A,VOLUME=(BAND1,BAND2),DEVICE=T9P,FSEQ=3
```

Der Operand FSEQ=3 hat Vorrang gegenüber der Angabe in der MULTIPLE FILE TAPE CONTAINS-Klausel.

Beispiel 32: Zuweisung verschiedener Eingabedateien bei nur einer Dateierklärung

```
/PASSWORD (C'EIN',X'FADE')      ①
/FILE MEIER.1,LINK=EIN          ②
/EXEC PROGR
/FILE MEIER.2,LINK=EIN          ③
/EXEC PROGR
/RELEASE EIN                    ④
/EXEC PROGR                     ⑤
```

① Die Kennwörter C'EIN' und X'FADE' werden dem System mitgeteilt, um den Zugriff zu geschützten Daten zu ermöglichen.

② Dem Dateikettungsnamen (= COBOL-Dateinamen) EIN wird die katalogisierte Datei MEIER.1 zugeordnet. Der Lademodul PROGR liest diese Datei, denn er enthält SELECT EIN ASSIGN DA-590-S-SYS010.

③ Dem Kettungsnamen EIN wird eine neue Datei zugeteilt. Sie wird anschließend von PROGR bearbeitet.

④ Die Verkettung von EIN und MEIER.2 wird aufgehoben.

⑤ In der Prozeßdateitabelle (TFT) gibt es keinen Eintrag für den Kettungsnamen EIN mehr. Das Programm PROGR versucht daher, aus einer Datei namens EIN (= COBOL-Dateiname) zu lesen.

5.2 Programmaufruf und Programmbeendigung

Der Benutzer möchte, nachdem er die evtl. benötigten Betriebsmittel vereinbart hat, nun sein COBOL-Programm ablaufen lassen.
Dies geschieht mit Hilfe des EXECUTE-Kommandos (Laden und Starten) oder des LOAD-Kommandos (Laden) und RESUME-Kommandos (Starten). Das LOAD-Kommando setzt man ein, wenn man vor dem Start noch andere Kommandos eingeben will, z. B. Testhilfe-Kommandos.

Aus der Form des Operanden im EXECUTE- oder LOAD-Kommando geht hervor, welcher Lader erforderlich ist : Liegt das Programm als Bindemodul vor, d. h. muß es noch gebunden werden, wird der Dynamische Bindelader (DLL) aufgerufen. Handelt es sich um einen Lademodul, d. h. ein gebundenes, ablauffähiges Programm, so wird der Lader des Systems angesprochen.

Bild 5-4 zeigt die verschiedenen Formen, in denen das COBOL-Programm vorliegen kann.

- als Bindemodul in der temporären Bindemoduldatei * des Prozesses,
- als Bindemodul in einer katalogisierten Bindemodulbibliothek,
- als Lademodul.

Das Beendigungsverhalten des Programmes ist insbesondere dann von Bedeutung, wenn es in einer Prozedur aufgerufen oder von einer Jobvariable überwacht wird.
Beim Auftreten von Fehlermeldungen, denen ein interner Return-Code zugeordnet ist (siehe dazu auch Fehlermeldung 9040 im Anhang), wird dieser Return-Code in die letzten beiden Bytes der Rückkehrcode-Anzeige einer überwachenden Jobvariable (sh. [22]) übernommen.
Die folgende Tabelle gibt einen Überblick über die möglichen Inhalte der Rückkehrcode-Anzeige und die zugeordneten Fehlermeldungen sowie deren Auswirkung auf den weiteren Ablauf einer Prozedur:

Rückkehrcode-Anzeige in Jobvariablen[1])	zugeordnete Fehlermeldungen (Nummer)[2])	Beendigung	Dump	Verhalten in Prozeduren
0100 0111[3])	keine	normal	nein	keine Verzweigung
2141	9082 9088	abnormal	nein	Verzweigung zum nächsten STEP-, ABEND-, ABORT- oder LOGOFF-Kommando
2142	9044			
2143	9074A			
2144	9101			
2145	9102			
2146	9038			
2151	9067	abnormal	nein	
2152	9056			
2153	9011			
2154	9068 9069 9070 9071 9076			
2155	9055			
2156	9057			
2157	9037			
3162	9026	abnormal	ja	
3163	9096			
3169	WROUT-Fehler: Es kann keine Meldung mehr ausgegeben werden			

[1]) Die beiden letzten Ziffern (fett gedruckt) stellen den internen Return-Code dar.
[2]) Inhalt und Bedeutung der Meldungen siehe Anhang 1.
[3]) Anwender-Return-Code (Users Return Code) ist gesetzt

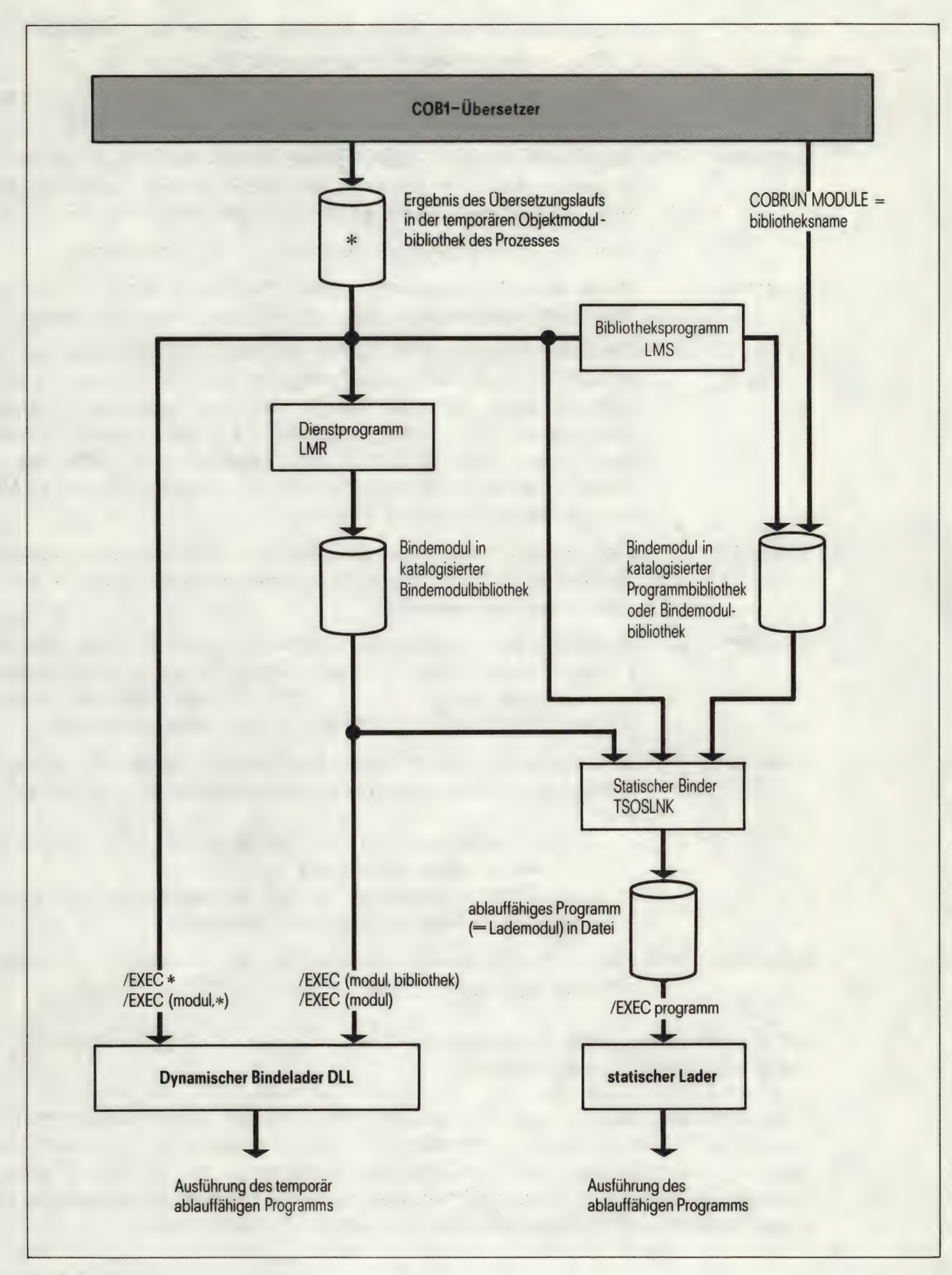


Bild 5-3

Möglichkeiten des Programmaufrufs

An die Stelle des EXEC-Kommandos können auch LOAD- und RESUME-Kommando (bzw. %RESUME-Kommando beim Testen mit AID) treten.

Das betreffende Format des EXECUTE-Kommandos [2] lautet:

Operation	Operanden
{EXECUTE | EXEC}	{* | (modul[,bibliothek]) | program}[, {IDA = {YES | <u>NO</u>} | SYMTEST = {ALL | <u>NO</u>}}][,MONJV = jvname]

*	bezeichnet die temporäre Bindemoduldatei des Prozesses, in die der COB1-Übersetzer das Objektprogramm ausgibt.
modul	gibt den Namen eines Bindemoduls (1 bis 8 Zeichen) an, der aus dem PROGRAM-ID-Namen des Quellprogramms gebildet wird.
bibliothek	Name einer Bindemodulbibliothek; dabei kann es sich um die temporäre Bindemoduldatei * oder um eine katalogisierte Datei handeln, die mit Hilfe des Dienstprogramms LMR oder LMS erstellt wurde. Fehlt die Angabe der Bibliothek, wird TASKLIB genommen.
program	Name einer katalogisierten Datei, die mit Hilfe des statischen Binders TSOSLNK erstellt wurde und in der sich ein Lademodul befindet. Das Dienstprogramm TSOSLNK bezieht den Bindemodul aus der temporären Bindemoduldatei des Prozesses oder einer katalogisierten Bibliothek. Abgelegt wurde der Modul entweder direkt vom COB1-Übersetzer durch die COBRUN-Anweisung MODULE=bibliotheksname oder mit Hilfe des Dienstprogramms LMS in einer katalogisierten LMS-Programm- oder Bindemodulbibliothek oder mit Hilfe des Dienstprogramms LMR in einer katalogisierten Bindemodulbibliothek.
IDA=YES	bewirkt, daß (falls vorhanden) das Internadreßbuch des Programms berücksichtigt wird, so daß in DTH-Kommandos symbolische Adressen verwendet werden können.
SYMTEST = ALL	ermöglicht es dem Benutzer, ohne weitere Vorkehrungen beim Testen mit AID symbolische Namen aus dem Quellprogramm zu verwenden. Voraussetzung dafür ist, daß es mit COBRUN SYMTEST=ALL übersetzt (und ggf. mit PROGRAM . . . ,SYMTEST = ALL gebunden) wurde.
SYMTEST = <u>NO</u>	ist Standardwert. Beim Testen des Programmes mit AID können nur dann symbolische Namen angegeben werden, wenn die zugehörigen Bindemoduln – bei der Übersetzung durch COBRUN SYMTEST = ALL mit LSD-Informationen versorgt wurden und – in einer PLAM-Bibliothek für das Nachladen der LSD-Informationen durch AID zur Verfügung stehen (siehe [25]).
MONJV=jvname	erklärt die Jobvariable mit dem Namen jvname zur programmüberwachenden Jobvariable (siehe [2], [22]).

Mit Ausnahme von MONJV stimmen die Operanden des LOAD-Kommandos [2] mit denen des EXECUTE-Kommandos überein.

Unter dem Laden eines Programms versteht man im BS2000 seine Übernahme aus einer Datei auf die Träger des virtuellen Adreßraums, d. h. die Gesamtheit von Haupt- und Seitenspeicher. Während des Programmlaufs stehen automatisch nur die Teile („Seiten", je 4096 Byte) des Programms im Hauptspeicher, die gerade zum Ablauf benötigt werden; ein Abbild dieser Seiten und alle übrigen befinden sich auf dem Seitenspeicher.

Hinweis:

Es ist nicht erlaubt, COBOL-Programme als Klasse 1-Programme zu laden.

5.3 Dialogtesthilfe AID

5.3.1 Einführung

Auch wenn COB1 ein COBOL-Programm als syntaktisch einwandfrei meldet, enthält es möglicherweise noch Logikfehler und läuft daher nicht in der gewünschten Weise ab. Für das Auffinden und Beseitigen solcher Fehler stehen dem COBOL-Programmierer verschiedene Hilfsmittel zur Verfügung:

- Er kann bereits in das Quellprogramm Testhilfezeilen einbauen und sie bei Bedarf aktivieren. Dies setzt voraus, daß schon bei der Erstellung des Quellprogrammes mögliche Fehlersituationen eingeplant werden und hat zur Folge, daß es zur Diagnose unvorhergesehener Fehler erforderlich werden kann, Testhilfezeilen abzuändern oder hinzuzufügen und anschließend das Quellprogramm neu zu übersetzen. Testhilfezeilen werden in [1] beschrieben.
- Er kann während des Programmlaufes die Dialogtesthilfe AID (**A**dvanced **I**nteractive **De**bugger) einsetzen. Sie erfordert keine Vorkehrungen bei der Programmierung und erlaubt es, im geladenen Programm während dessen Ausführung Fehler zu suchen und vorübergehend zu beheben.

In diesem Benutzerhandbuch soll AID lediglich kurz vorgestellt werden. Die ausführliche Beschreibung dieser Testhilfe findet sich in [25].

AID zeichnet sich durch folgende Leistungsmerkmale aus:

1. Es bietet die Möglichkeit, „symbolisch" zu testen, d. h. in den Kommandos anstelle sedezimaler Adressen auch symbolische Namen aus dem Quellprogramm anzugeben, wenn die dafür nötigen LSD-Informationen beim Übersetzen erzeugt und später an das geladene Programm weitergegeben werden (siehe 5.3.2).
   Dabei ist es nicht unbedingt erforderlich, diese Informationen stets für das Gesamtprogramm zusammen mit diesem Programm zu laden. AID erlaubt nämlich ein Nachladen der LSD-Informationen für jede Übersetzungseinheit, falls die zugehörigen Bindemoduln (mit den LSD-Informationen) in einer PLAM-Bibliothek stehen. Dadurch lassen sich Betriebsmittel wirtschaftlicher einsetzen:
   - Der Programmspeicher wird entlastet, da LSD-Informationen nur dann geladen werden müssen, wenn sie zum Testen benötigt werden (der Speicherbedarf für ein Programm steigt durch das Mitladen dieser Informationen ungefähr auf das Fünffache).
   - Ein Programm, das im Test fehlerfrei bleibt, muß für den Produktiveinsatz nicht unbedingt neu (ohne LSD-Informationen) übersetzt oder gebunden werden.
   - Falls sich für ein Programm während seines Produktiveinsatzes ein Test als nötig erweist, stehen dafür LSD-Informationen zur Verfügung, ohne daß das Programm erneut übersetzt und gebunden werden muß.
2. Es stellt Funktionen zur Verfügung, die es insbesondere gestatten,
   - den Programmablauf auf symbolischer Ebene zu verfolgen und zu protokollieren (TRACE-Funktion)
   - den Programmablauf an festgelegten Stellen oder beim Eintreten definierter Ereignisse zu unterbrechen, um AID- oder BS2000-Kommandos (sogenannte Subkommandos) ausführen zu lassen
   - sich die Inhalte von Feldern in einer Form ausgeben zu lassen, welche die Datendefinitionen des Quellprogrammes berücksichtigt
   - die Inhalte von Feldern zu verändern, wobei AID die dazu nötigen Datenübertragungen gemäß den Regeln der COBOL-MOVE-Anweisung durchführt

3. Es unterstützt neben der Diagnose geladener Programme auch die Analyse von Speicherabzügen in Plattendateien.
4. Es kann außer im Dialog- auch im Stapelbetrieb eingesetzt werden. Für einen Programmtest empfiehlt sich allerdings der Dialog, da die Folge der Kommandos nicht im voraus festgelegt werden muß und der jeweiligen Testsituation angepaßt werden kann.

5.3.2 Voraussetzungen für das symbolische Testen

Beim Testen auf symbolischer Ebene erlaubt es AID, Datenfelder, Kapitel und Paragraphen mit den im Quellprogramm definierten Namen anzusprechen und sich auf Anweisungszeilen und einzelne COBOL-Verben in der PROCEDURE DIVISION zu beziehen. Dafür müssen AID Informationen über diese symbolischen Namen zur Verfügung gestellt werden. Diese Informationen gliedern sich in zwei Teile (siehe dazu [23]),

- die LSD (List for Symbolic Debugging), in der die im Modul definierten symbolischen Namen und Anweisungen verzeichnet sind und
- das ESD (External Symbol Dictionary), das die Externbezüge eines Moduls registriert.

Die Erzeugung bzw. Weitergabe dieser Informationen wird bei jedem der folgenden Schritte

- Übersetzen mit COB1
- Binden und Laden mit DLL oder
- Binden mit TSOSLNK
- Laden mit ELDE

durch den Operanden SYMTEST veranlaßt oder unterdrückt. Dabei werden ESD-Informationen standardmäßig generiert und weitergegeben, während die LSD-Informationen AID auf zwei Wegen zugänglich gemacht werden können: Nachdem sie bei der Übersetzung erzeugt worden sind, ist es möglich,

- sie zusammen mit dem Gesamtprogramm zu laden oder
- sie erst bei Bedarf für jede Übersetzungseinheit nachzuladen, falls die zugehörigen Bindemoduln in einer PLAM-Bibliothek stehen.

Die folgende Tabelle gibt für beide Fälle einen Überblick über die Werte, die dem SYMTEST-Operanden bei jedem Schritt zugeordnet werden müssen (zur genaueren Information siehe [25]).

Schritte in der Programmentwicklung	Werte der SYMTEST-Operanden,	
	wenn die LSD-Information zusammen mit dem Gesamtprogramm geladen werden soll	wenn später die LSD-Information durch AID nachgeladen werden soll[1])
Übersetzen mit COB1	COBRUN SYMTEST = ALL	COBRUN SYMTEST = ALL
Binden und Laden mit DLL	LOAD . . .,SYMTEST = ALL oder EXEC . . .,SYMTEST = ALL	LOAD . . . . [,SYMTEST = NO] oder EXEC . . . [,SYMTEST = NO]
Binden mit TSOSLNK	PROGRAM . . .,SYMTEST = ALL	PROGRAM . . . [,SYMTEST = MAP]
Laden mit ELDE	LOAD . . .,SYMTEST = ALL oder EXEC . . .,SYMTEST = ALL	LOAD . . .[,SYMTEST = NO] oder EXEC . . .[,SYMTEST = NO]

[1]) Dies ist nur dann möglich, wenn die zugehörigen Bindemoduln in einer PLAM-Bibliothek stehen

5.3.3 Symbolisches Testen mit AID

Beim symbolischen Testen mit AID können Datenfelder, Kapitel und Paragraphen mit den Namen angesprochen werden, die im Quellprogramm definiert wurden.

Um dagegen auf eine Zeile in der PROCEDURE DIVISION Bezug zu nehmen, muß man einen Namen der Form

- S'nnnnn' (für eine Zeile ohne COBOL-Verb) bzw.
- S'nnnnnverbk' (für eine Zeile mit COBOL-Verben)

angeben. Einen solchen **LSD-Namen** bildet COB1 für jede Zeile in der PROCEDURE DIVISION und für jedes COBOL-Verb in einer Anweisungszeile. Seine Bestandteile haben dabei folgende Bedeutung:

nnnnn ist die maximal fünfstellige Nummer dieser Zeile in der PROCEDURE DIVISION, die COB1 bei der Übersetzung vergeben hat. Sie muß ohne führende Nullen angegeben werden.

verb ist die festgelegte Abkürzung eines COBOL-Verbs in der betreffenden Zeile. Diese Abkürzungen können der nachstehenden Liste entnommen werden.

k ist eine einstellige Nummer, die angibt, das wievielte von mehreren gleichen COBOL-Verben innerhalb der Zeile nnnnn bezeichnet werden soll.
Falls k gleich 1 ist, wird es weggelassen.

Liste der Abkürzungen für COBOL-Verben:

ACC	ACCEPT	SEA	SEARCH
ADD	ADD	SET	SET
ADDC	ADD CORRESPONDING	SOR	SORT
ALT	ALTER	STA	START
CALL	CALL	STO	STOP
CANC	CANCEL	STOR	STORE
CLO	CLOSE	STRG	STRING
COM	COMPUTE	SUB	SUBTRACT
CON	CONNECT	SUBC	SUBTRACT CORRESPONDING
DEL	DELETE	TER	TERMINATE
DIS	DISPLAY	UNST	UNSTRING
DIV	DIVIDE	WRI	WRITE
DSC	DISCONNECT		
ENTR	ENTRY		
ERA	ERASE		
EXI	EXIT		
EXIT	EXIT PROGRAM		
FET	FETCH		
FIN	FINISH		
FND	FIND		
FRE	FREE		
GEN	GENERATE		
GET	GET		
GOT	GO␣TO		
IF	IF		
INI	INITIATE		
INIT	INITIALIZE		
INSP	INSPECT		
KEE	KEEP		
MOD	MODIFY		
MOV	MOVE		
MOVC	MOVE CORRESPONDING		
MRG	MERGE		
MUL	MULTIPLY		
ON	ON		
OPE	OPEN		
PER	PERFORM		
REA	READ		
REDY	READY		
REL	RELEASE		
RET	RETURN		
REW	REWRITE		

Beispiel 33: Bildung von LSD-Namen

```
000026␣␣␣␣␣␣␣␣␣IF A = B MOVE A TO D MOVE B TO E.
```

In dieser Anweisungszeile hat
- das erste Verb den LSD-Namen S'26IF',
- das zweite Verb den LSD-Namen S'26MOV' und
- das dritte Verb den LSD-Namen S'26MOV2'

Beispiel 34: Testlauf mit AID-Kommandos
Hinweis: Die AID-Kommandos sind ausführlicher in [25] beschrieben

```
/PARAM CODE=2,DIAG=NO,MAP=NO
/EXEC $COB1
%  BLS0500 PROGRAM 'COB1', VERSION '23A' OF '86-04-24' LOADED.
*COBRUN SYMTEST=ALL,MODULE=PLAM.LIB                                     (1)
9099 COBRUN SYMTEST=ALL,MODULE=PLAM.LIB
*END QUELL.EINXEINS
9099 END QUELL.EINXEINS
9017 COMPILATION INITIATED, VERSION IS  V02.3A
9097 COMPILATION COMPLETED WITHOUT ERRORS
9004 COMPILATION OF EINXEINS USED  01,29  CPU SECONDS
/LOAD (EINXEINS,PLAM.LIB)                                               (2)
%  BLS0001 DLL VER 823
%  BLS0517 MODULE 'EINXEINS' LOADED
/%TRACE 10                                                              (3)
I378 SYMBOLIC INFORMATION MISSING
/%SYMLIB PLAM.LIB                                                       (4)
/%TRACE 10
    18DIS                          I-O-ACCESS
ZWEISTELLIGE ZAHL EINGEBEN (ENDE BEI 0):
    20ACC                          I-O-ACCESS
*17
    21IF                           IF
    24IF                           IF
    25PER                          CALL                                 (5)
    28MUL                          ASSIGN
    29DIS                          I-O-ACCESS
01*17= 17
    28MUL                          ASSIGN
    29DIS                          I-O-ACCESS
02*17= 34
    28MUL                          ASSIGN

  STOPPED AT SRC_REF: 28MUL , SOURCE: EINXEINS, PROC: EINXEINS

/%INSERT RECHNEN <%DISPLAY 'RECHNEN',I,ZAHL,ERGEBNIS,ELEMENT (I - 1)> ONY 3 S   (6)
/%RESUME
03*17= 51
RECHNEN
CURRENT TASK *** TSN: 8521 *****************************************************
SRC_REF:      28MUL   SOURCE: EINXEINS  PROC: EINXEINS  ************************
I                =    4
ZAHL             =   17
ERGEBNIS         = | 51|
ELEMENT(3)       =    51
04*17= 68
RECHNEN
I                =    5
ZAHL             =   17                                                 (7)
ERGEBNIS         = | 68|
ELEMENT(4)       =    68
05*17= 85
RECHNEN
I                =    6
ZAHL             =   17
ERGEBNIS         = | 85|
ELEMENT(5)       =    85

   STOPPED AT SRC_REF: 28MUL , SOURCE: EINXEINS  PROC: EINXEINS
```

5

```
/ %SET ERGEBNIS INTO ZAHL                                                   } (8)
I388 TYPES ARE NOT CONVERTIBLE. NOTHING CHANGED ( CMD: SET      )           }
/ %SET 85 INTO ZAHL                                                           (9)
/ %INSERT ANFANG <%DISPLAY 'ANFANG';%SDUMP;%RESUME>                            (10)
/ %RESUME
06*85=510
07*85=595
08*85=680
09*85=765
10*85=850
ANFANG
SRC_REF:      18DIS   SOURCE: EINXEINS  PROC: EINXEINS  ************************
ZERO            =           0

HIGH-VALUE      = FF   .

LOW-VALUE       = 00   .

SPACE           = | |

QUOTE           = |"|

ZAHL            =   85

ERGEBNIS        = |850|                                                       (11)

I               =   11

01          ERGEBNISTABELLE
 05         ELEMENT(1:10)
            ( 1)     17  ( 2)     34  ( 3)     51  ( 4)     68  ( 5)     85
            ( 6)    510  ( 7)    595  ( 8)    680  ( 9)    765  (10)    850

TODAYS-DATE     = |05/14/86134|

CURRENT-DATE    = |05/14/86134|
TALLY           =       +0

RETURN-CODE     =      +0
ZWEISTELLIGE ZAHL EINGEBEN (ENDE BEI 0):
*00
```

① COB1 wird angewiesen,
- zusätzlich zu den ESD- auch LSD-Informationen zu erzeugen und
- den Bindemodul in eine PLAM-Bibliothek zu schreiben (um später ein Nachladen der LSD-Informationen zu ermöglichen)

② Der Bindemodul EINXEINS wird zunächst ohne LSD-Informationen gebunden und geladen

③ Es wird versucht, den Programmablauf auf symbolischer Ebene zu verfolgen. Wegen fehlender LSD-Informationen weist AID das Kommando ab.

④ Das %SYMLIB-Kommando weist AID an, die fehlenden LSD-Informationen aus der PLAM-Bibliothek nachzuladen, in der sich der Bindemodul befindet.

⑤ AID akzeptiert das %TRACE-Kommando und protokolliert den Programmablauf während der Ausführung von zehn Anweisungen. Jede Protokollzeile besteht aus
- der von COB1 vergebenen Nummer der Anweisungszeile, gefolgt von
- der Abkürzung des dazugehörigen COBOL-Verbs, sowie
- einem Kommentar zum Anweisungstyp.

Ein-/Ausgaben über Terminal, die das Programm selbst veranlaßt, werden jeweils im Anschluß an die Protokollzeilen angezeigt, welche die zugehörigen Ein-/Ausgabeanweisungen enthalten.
Nach der Beendigung der Ablaufverfolgung wird das Programm unterbrochen, und es können weitere AID-Kommandos eingegeben werden.

⑥ An den Anfang des Paragraphen RECHNEN wird ein Haltepunkt gesetzt. Vor der Ausführung der ersten Anweisungen dieses Paragraphen soll AID den Programmlauf unterbrechen und das vereinbarte Subkommando ausführen. In diesem Fall weist das Subkommando AID an, das Wort „RECHNEN" und die Inhalte der Felder I, ZAHL, ERGEBNIS sowie des I—1-ten Elementes der ERGEBNISTABELLE auszugeben.
Nach drei Durchläufen soll AID den Haltepunkt wieder löschen und den Programmlauf unterbrechen.

⑦ Jedesmal beim Erreichen des Haltepunktes werden die Inhalte der Felder I, ZAHL, ERGEBNIS und ELEMENT (I—1) zusammen mit den symbolischen Feldnamen ausgegeben. Dabei werden die zugehörigen Definitionen in der DATA DIVISION des Quellprogrammes berücksichtigt.

⑧ Es wird versucht, den Inhalt des (numerisch druckaufbereiteten) Feldes ERGEBNIS in das (numerische) Feld ZAHL zu übertragen. Dies ist nach den Regeln der COBOL-MOVE-Anweisung unzulässig, und AID weist das Kommando mit einer Fehlermeldung ab.

⑨ In das Feld ZAHL wird das numerische Literal 85 übernommen.

⑩ Bei ANFANG wird ein Haltepunkt gesetzt. In der zugehörigen Subkommandofolge werden mehrere Kommandos miteinander verkettet. Sie weisen AID an, beim Erreichen des Haltepunktes
- das Wort „ANFANG" auszugeben,
- einen symbolischen Dump der gesamten DATA DIVISION zu erzeugen und anschließend
- den Programmlauf wieder fortzusetzen.

⑪ Beim symbolischen Dump bereitet AID alle in der DATA DIVISION vereinbarten Datenfelder gemäß ihrem Typ und ihrer Struktur auf und gibt sie zusammen mit ihren aktuellen Werten aus, ebenso die COBOL-Sonderregister und die figurativen Konstanten.

5.4 Prozeßschalter und Benutzerschalter

Das BS2000 bietet 32 Prozeßschalter (0 bis 31) und 32 Benutzerschalter (0 bis 31) (siehe Manual „Kommandosprache", [2]). Der Benutzer verwendet diese Schalter

- auf Betriebssystem-Ebene
- auf COBOL-Programm-Ebene.

Prozeßschalter können verwendet werden, wenn sich mehrere COBOL-Programme innerhalb eines Prozesses verständigen sollen, d.h. z.B. wenn der Ablauf eines Programms von Verarbeitungsschritten eines anderen Programms abhängt. Wenn sich dagegen verschiedene Prozesse verständigen sollen, sind Benutzerschalter zu verwenden.

Ein COBOL-Programm kann den Status von Prozeß- oder Benutzerschaltern

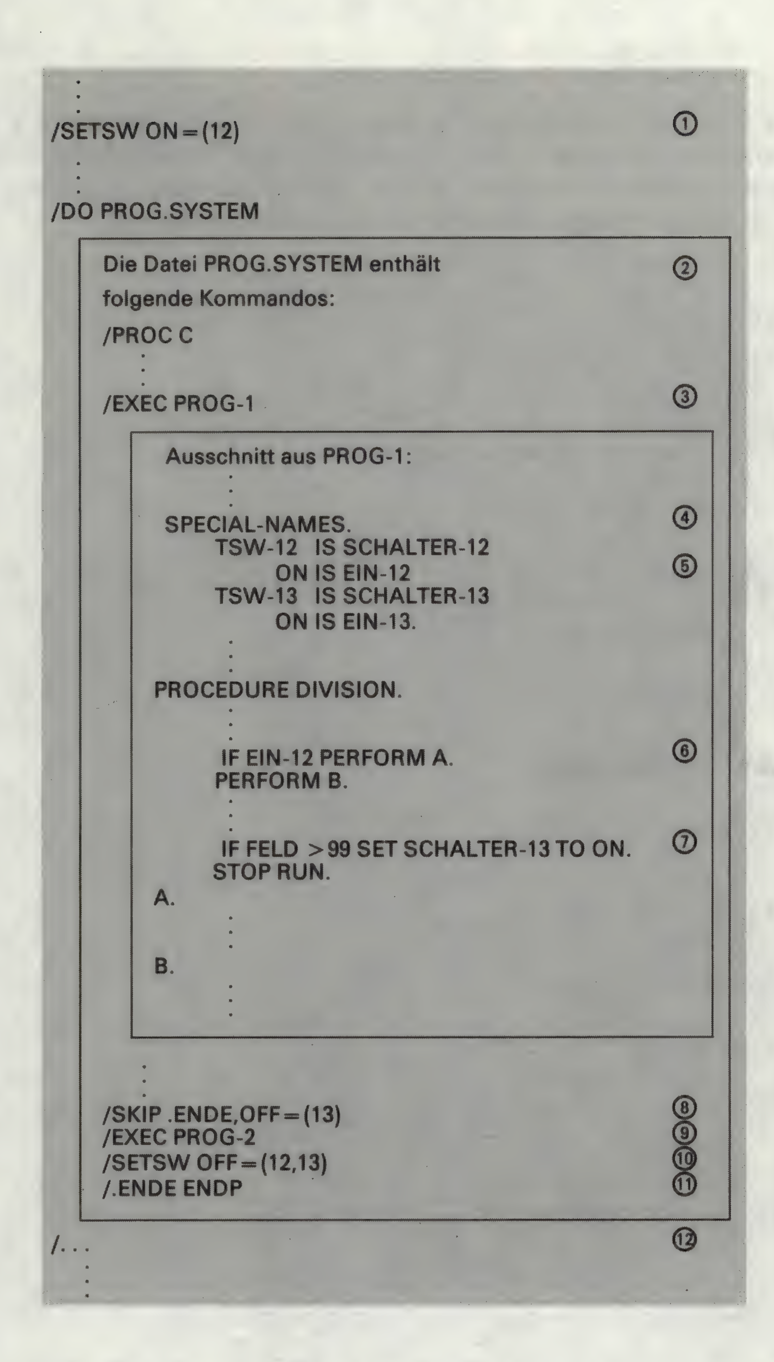
- verändern
- abfragen.

Folgende Schlüsselwörter stehen zur Verfügung (siehe Manual „COB1 Beschreibung", [1]):

- für die Prozeßschalter TSW-0, ... TSW-31;
- für die Benutzerschalter USW-0, ... USW-31.

Beispiel 34a: Verwendung eines Prozeßschalters

Eine DO-Prozedur sieht verschiedene Verarbeitungsvarianten vor, abhängig vom Status der Prozeßschalter 12 und 13. Sie werden sowohl auf Betriebssystem-Ebene als auch auf Programm-Ebene bedient: Prozeßschalter 12 kann zunächst auf Betriebssystem-Ebene gesetzt werden, um die Verarbeitung innerhalb der folgenden DO-Prozedur zu steuern. Dort wird auf Programm-Ebene sein Zustand abgefragt; bei bestimmter Programmverarbeitung wird Prozeßschalter 13 gesetzt, der anschließend auf Betriebssystem-Ebene abgefragt wird. Ausschnitt aus dem Dialogprozeß:



```
 :
/SETSW ON=(12)                                        ①
 :
/DO PROG.SYSTEM
   Die Datei PROG.SYSTEM enthält                      ②
   folgende Kommandos:
   /PROC C
     :
   /EXEC PROG-1                                       ③
       Ausschnitt aus PROG-1:
           :
       SPECIAL-NAMES.                                 ④
           TSW-12  IS SCHALTER-12
                ON IS EIN-12                          ⑤
           TSW-13  IS SCHALTER-13
                ON IS EIN-13.
           :
       PROCEDURE DIVISION.
           :
            IF EIN-12 PERFORM A.                      ⑥
           PERFORM B.
           :
            IF FELD >99 SET SCHALTER-13 TO ON.        ⑦
           STOP RUN.
       A.
           :
       B.
           :
     :
   /SKIP .ENDE,OFF=(13)                               ⑧
   /EXEC PROG-2                                       ⑨
   /SETSW OFF=(12,13)                                 ⑩
   /.ENDE ENDP                                        ⑪
/...                                                  ⑫
 :
```

① Prozeßschalter 12 erhält den Status ON auf Betriebssystem-Ebene.

② Ausschnitt aus einer DO-Prozedur.

③ Das COBOL-Programm PROG-1 (Lademodul) soll ablaufen.

④ Im SPECIAL-NAMES-Paragraphen wird der Prozeßschalter 12 mit TSW-12 vereinbart und die Beziehung zu Benutzerwörtern hergestellt.

⑤ Das Benutzerwort SCHALTER-12 wird verwendet, wenn der Prozeßschalter 12 verändert werden soll; das Benutzerwort EIN-12 dient zur Abfrage. Entsprechendes gilt für den Prozeßschalter 13.

⑥ Der ON-Status wird abgefragt, PERFORM A in Abhängigkeit von ① vor PERFORM B ausgeführt.

⑦ Prozeßschalter 13 erhält auf Programm-Ebene den Status ON, falls eine bestimmte Verarbeitungsart — angezeigt durch den Inhalt von FELD — abgeschlossen wurde.

⑧ Verzweigung zum Prozedurende, falls PROG-1 den Prozeßschalter 13 nicht gesetzt hat; siehe ⑦.

⑨ Andernfalls wird zusätzlich zum PROG-1 PROG-2 ausgeführt.

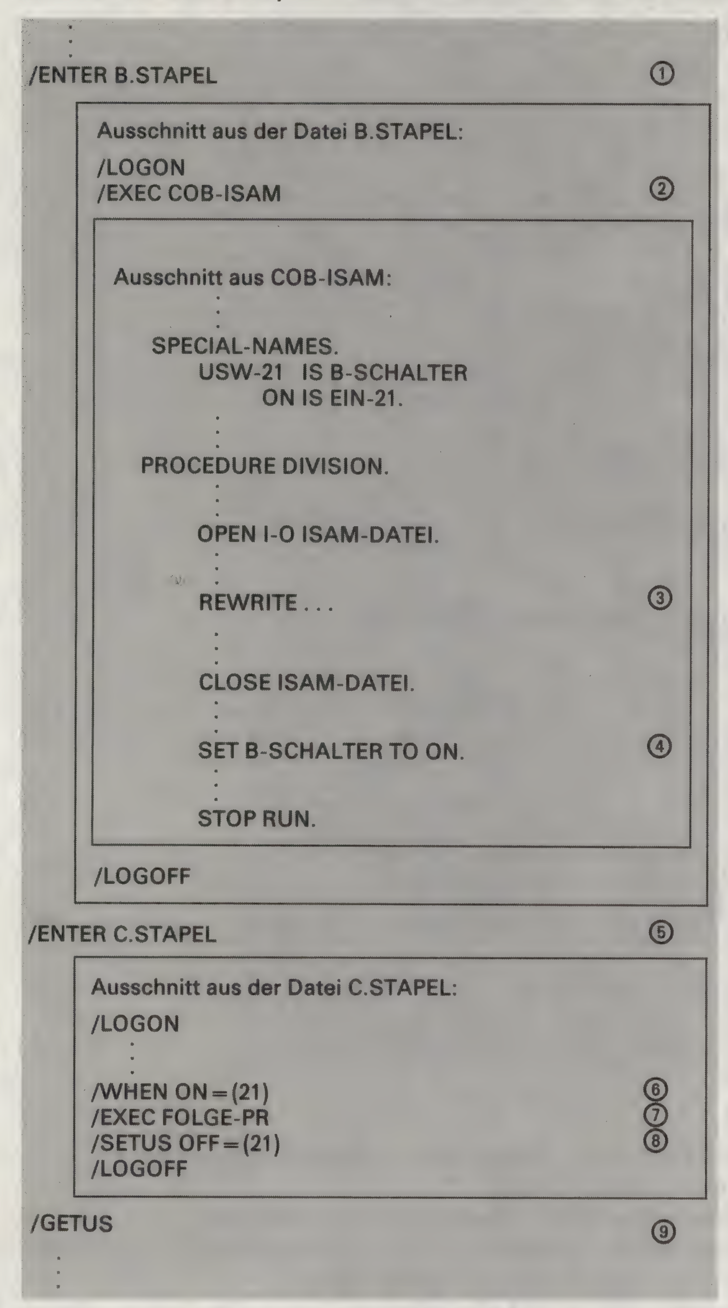
⑩ Rücksetzen beider Prozeßschalter.

⑪ Ende der DO-Prozedur.

⑫ Fortsetzung des Dialogbetriebs.

Beispiel 34b: Verwendung eines Benutzerschalters

Der folgende Dialogprozeß A erzeugt zwei Stapelprozesse B und C. Im Prozeß B wird eine ISAM-Datei aktualisiert. Erst danach kann Prozeß C ablaufen. Benutzerschalter 21 wird in drei verschiedenen Prozessen verwendet. Einmal wird er auf Programm-Ebene bedient, zweimal auf Betriebssystem-Ebene. Ausschnitt aus dem Dialogprozeß A:



```
  :
/ENTER B.STAPEL                                        ①

    Ausschnitt aus der Datei B.STAPEL:
    /LOGON
    /EXEC COB-ISAM                                     ②

      Ausschnitt aus COB-ISAM:
             :
        SPECIAL-NAMES.
            USW-21   IS B-SCHALTER
                 ON IS EIN-21.
             :
       PROCEDURE DIVISION.
             :
           OPEN I-O ISAM-DATEI.
             :
           REWRITE . . .                               ③
             :
           CLOSE ISAM-DATEI.
             :
           SET B-SCHALTER TO ON.                       ④
             :
           STOP RUN.

    /LOGOFF
/ENTER C.STAPEL                                        ⑤

    Ausschnitt aus der Datei C.STAPEL:
    /LOGON
      :
    /WHEN ON=(21)                                      ⑥
    /EXEC FOLGE-PR                                     ⑦
    /SETUS OFF=(21)                                    ⑧
    /LOGOFF

/GETUS                                                 ⑨
  :
```

① Mit der Datei B.STAPEL (ENTER-Datei) wird der Stapelprozeß B erzeugt.
② Das COBOL-Programm COB-ISAM (Lademodul) soll ablaufen.
③ Die Datei ISAM-DATEI wird aktualisiert.
④ Der Status ON des Benutzerschalters 21 markiert das Ende der Aktualisierung (im Prozeß B).
⑤ Mit der Datei C.STAPEL (ENTER-Datei) wird der Stapelprozeß C erzeugt.
⑥ Prozeß C wartet solange, bis im Prozeß B der Benutzerschalter 21 den Status ON erhält (siehe auch Manual „Kommandosprache", [2]).
⑦ Das COBOL-Programm FOLGE-PR soll ablaufen; es benötigt die im Prozeß B aktualisierte ISAM-Datei.
⑧ Benutzerschalter 21 erhält den Status OFF, um das Ende von Prozeß C zu markieren.
⑨ Der Status des Benutzerschalters 21 wird im Dialogprozeß A abgefragt.

5.5 Verwendung von Jobvariablen

Die „Jobvariablen" bilden ein eigenes Softwareprodukt (ab BS2000 V7.1). Sie sind im Manual „Jobvariablen" [22] beschrieben. — Hier in diesem Abschnitt werden speziell dem COBOL-Programmierer Anwendungsmöglichkeiten für Benutzerjobvariablen aufgezeigt.

Eine Jobvariable (JV) gestattet Informationsaustausch zwischen

— COBOL-Programm und Kommandosprache
— mehreren Programmen.

Ein COBOL-Programm liest eine JV mit der Anweisung ACCEPT, beschreibt/verändert eine JV mit der Anweisung DISPLAY (siehe COB1 Beschreibung). Die Anweisungen ACCEPT und DISPLAY müssen sich auf Merknamen beziehen, die im SPECIAL-NAMES-Paragraphen der CONFIGURATION SECTION (ENVIRONMENT DIVISION) mit Funktionsnamen verknüpft sein müssen. Diese Funktionsnamen lauten

JV-linkname

Für „linkname" vergibt der Programmierer höchstens 7 Zeichen. Auf Kommandoebene wird daraus der Linkname „*linkname" gebildet. Dieser wird mit dem Katalognamen der JV verbunden; Kommando /DCLJV jvname,LINK=*linkname.

Eine JV muß entweder mit dem Kommando /CATJV oder mit /DCLJV eingerichtet werden. Weitere Kommandos sind /SETJV (Beschreiben), /ERAJV (Löschen), /GETJV (Lesen) u. a.

Ein COBOL-Programm kann immer nur auf den gesamten Inhalt einer JV, nicht aber auf Teilzeichenfolgen zugreifen — im Gegensatz zu Kommandos. DISPLAY verändert (bzw. beschreibt) den Inhalt einer JV in der Länge des Ausgabefeldes. Unterstützt das Betriebssystem Jobvariablen nicht (also BS2000 ohne JV), so verursacht der erste Zugriffsversuch die Meldung 9082 (SYSOUT); das ACCEPT-Feld enthält dann /* ab Spalte 1.

Unterstützt das Betriebssystem Jobvariablen, ist jedoch der Zugriff fehlerhaft, so kommt die Meldung 9088; wurde über eine ACCEPT-Anweisung zugegriffen, so enthält das ACCEPT-Feld danach ab Spalte 1 '/*'.

Beispiel: Kommunikation über Jobvariable

Im folgenden Dialogprozeß wird die JV KONTROLLE.ABLAUF sowohl von einem COBOL-Programm als auch auf Kommandoebene verwendet. Abhängig vom Inhalt der JV kann das Programm unterschiedliche Verarbeitungszweige durchlaufen und ggf. den Inhalt der JV aktualisieren. Auch ein anderer Prozeß — selbst unter einer anderen Benutzerkennung — könnte auf diese JV zugreifen, Voraussetzung dazu: /CATJV . . ., SHARE=YES.

```
/DCLJV KONTROLLE.ABLAUF, LINK=*AKTUELL                                  ①
/EXEC PROG.ARBEIT-1                                                      ②
```

Programmausschnitt:

```
        .
        .
        .
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.
    JV-AKTUELL IS FELDJV.                                                ③
        .
        .
        .
DATA DIVISION.
WORKING-STORAGE SECTION.
77  TAGDAT          PIC X(6).                                            ④
```

```
01  INHALT-JV.                                                  ⑤
  05  AKT-DAT       PIC X(6).                                   ⑥
  05  FILLER        PIC X(20).                                  ⑦
  05  AKT-NUM       PIC 9(4).                                   ⑧
          .
          .
          .
PROCEDURE DIVISION.
    ACCEPT INHALT-JV FROM FELDJV.                               ⑨
    ACCEPT TAGDAT FROM DATE.                                    ⑩
    IF AKT-DAT NOT EQUAL TAGDAT                                 ⑪
      PERFORM ARBEIT
        ELSE PERFORM SCHON-AKTUELL.
          .
          .
          .
ARBEIT.
          .
          .
          .
    MOVE TAGDAT TO AKT-DAT.                                     ⑫
    ADD 1 TO AKT-NUM.                                           ⑬
    DISPLAY INHALT-JV UPON FELDJV.                              ⑭
          .
          .
          .
SCHON-AKTUELL.
    DISPLAY "ENDE AKTUALISIERUNG"
      UPON TERMINAL.                                            ⑮
          .
          .
          .
 /GETJV KONTROLLE.ABLAUF                                        ⑯
%820926 AKTUALISIERUNG NR. 1679
          .
          .
          .
```

① Der JV-Katalogname KONTROLLE.ABLAUF wird mit dem Linknamen * AKTUELL verknüpft.
② Aufruf des COBOL-Programms.
③ Der Funktionsname JV-AKTUELL wird mit dem Merknamen FELDJV verknüpft.
④ Das Feld TAGDAT wird für das Tagesdatum reserviert.
⑤ Feldvereinbarungen für den Inhalt der JV:
⑥ AKT-DAT ist für das Datum vorgesehen.
⑦ Hier enthält die JV einen Text, den das COBOL-Programm nicht verwendet.
⑧ AKT-NUM ist für eine Kontrollnummer vorgesehen.
⑨ Einlesen der JV.
⑩ Einlesen des Tagesdatums.
⑪ Vergleich des Datums in der JV mit dem Tagesdatum:
Bei Gleichheit wird die Prozedur ARBEIT übersprungen, welche den Inhalt der JV aktualisiert.
⑫ Eintrag des Tagesdatums in die JV.
⑬ Hochzählen der Kontrollnummer.
⑭ Rückschreiben der JV.
⑮ Ausgabe auf dem Bildschirm zur Kontrolle des Programmablaufs.
⑯ Auf Kommandoebene kann die JV eingesehen werden: Datum und Nummer der letzten Aktualisierung sind darin vermerkt.

5.6 Zugriff auf Übersetzer- und Betriebssysteminformationen

Der COBOL-Programmierer kann mit Hilfe der ACCEPT-Anweisung über Informationen des Übersetzers und des Betriebssystems verfügen. Es sind Informationen über

die Übersetzung des Quellprogramms,
die seit dem LOGON-Kommando verbrauchte CPU-Zeit,
den Prozeß, unter dem das Objektprogramm abläuft, und
die Datenstation.

Durch den Eintrag

funktionsname IS merkname

im SPECIAL-NAMES-Paragraphen der ENVIRONMENT DIVISION und durch die Ausführung der Anweisung

ACCEPT bezeichner FROM merkname

werden die durch funktionsname angegebenen Informationen nach bezeichner übertragen. funktionsname muß dabei eines der folgenden Wörter sein:

COMPILER-INFO
CPU-TIME
PROCESS-INFO
TERMINAL-INFO

Die folgende Tabelle gibt für jeden Funktionsnamen Aufschluß darüber,

a) welche Informationen nach Ausführung der ACCEPT-Anweisung in bezeichner zur Verfügung stehen und

b) wie die Informationen auf die Zeichenpositionen des Bezeichners verteilt sind.

Tabelle: Informationsarten für die verschiedenen Funktionsnamen

Zeichenposition(en)	Art der Informationen für COMPILER-INFO
1–10	Version des Compilers Format: Vzz.zbzzzz z = Ziffer oder Leerzeichen, b = Buchstabe oder Leerzeichen, (z. B. „V01.2A “)
11–18	Datum der Übersetzung Format: JJ-MM-TT (z. B. „99-12-31“)
19–26	Uhrzeit der Übersetzung Format: HH-MM-SS (z. B. „23-59-59“)
27–56	Die ersten 8 Zeichen des PROGRAM-ID-Namens

Zeichenposition(en)	Art der Information für CPU-TIME
1–9	CPU-Zeit in Millisekunden

Zeichenposition(en)	Art der Informationen für PROCESS-INFO
1	Prozeßtyp Inhalt: B für Batch D für Dialog
2–5	Prozeßfolgenummer
6–13	Benutzerkennung
14–21	Abrechnungsnummer
22	Privilegierungszeichen des Prozesses Inhalt: U für Benutzer S für Systemverwalter
22–32	Betriebssystemversion Format: Vzz.zbzzzz (z. B. „V07.1 “)

Zeichenposition(en)	Art der Informationen für TERMINAL-INFO
1–8	Stationsname
9–13	Anzahl der Zeichen pro physikalische Zeile
14–18	Anzahl der physikalischen Zeilen, die ausgegeben werden können, ohne daß die Informationsüberlaufkontrolle anspricht.
19–23	Anzahl der Zeichen, die ausgegeben werden können, ohne daß die Informationsüberlaufkontrolle anspricht.
24–27	Gerätetyp (z. B. „8160“)

Beispiel: **Datenstruktur für Compiler-, Prozeß- und Terminal-Informationen bzw. CPU-Zeit, abrufbar durch ACCEPT-Anweisung:**

```
*     BEISPIEL EINER DATENSTRUKTUR FUER UEBERNAHME
*     DER INFORMATIONEN, DIE DURCH DAS NEUE FORMAT
*     DER ACCEPT ANWEISUNG ANGEBOTEN WIRD.
*
 01   COMPILER-INFORMATION.
      02 COMPILER-VERSION                          PIC X(10).
      02 UEBERSETZUNGS-DATUM                       PIC X(8).
      02 UEBERSETZUNGS-ZEIT                        PIC X(8).
      02 PROGRAM-NAME                              PIC X(30).
*
 01   CPU-ZEIT                                     PIC 9(9).
*
 01   PROZESS-INFORMATION.
      02 PROZESS-ART                               PIC X.
         88 BATCH-PROZESS                                    VALUE "B".
         88 DIALOG-PROZESS                                   VALUE "D".
      02 PROZESS-FOLGENUMMER                       PIC 9(4).
      02 BENUTZERKENNUNG                           PIC X(8).
      02 ABRECHNUNGSNUMMER                         PIC X(8).
      02 PRIVILEGIERUNGSKENNZEICHEN                PIC X.
         88 SYSTEMVERWALTER                                  VALUE "S".
         88 BENUTZER                                         VALUE "U".
      02 BETRIEBSSYSTEMVERSION                     PIC X(10).

*
 01   TERMINAL-INFORMATION.
      02 STATIONS-NAME                             PIC X(8).
      02 ZEICHEN-ZAHL-ZEILE                        PIC 9(5).
      02 ZEILEN-ZAHL-SCHIRM                        PIC 9(5).
      02 ZEICHEN-ZAHL-SCHIRM                       PIC 9(5).
      02 GERAETE-TYP                               PIC X(4).
```

5

Programmierhinweise

1. Die mit ACCEPT angeforderte Information wird unabhängig vom TYP von bezeichner übertragen. Sie wird abdruckbar, linksbündig und der Länge von bezeichner entsprechend abgelegt. Ist bezeichner zu lang, so werden die durch die ACCEPT-Anweisung nicht beschriebenen Zeichenpositionen von bezeichner mit Leerzeichen belegt. Ist bezeichner zu kurz, so werden nur so viele Zeichen übertragen, wie bezeichner Zeichenpositionen hat.

2. Die Informationen über die Datenstation sind selbstverständlich nur im Teilnehmerbetrieb sinnvoll.

   Der Gerätetyp bezeichnet in diesem Zusammenhang den für die Leitung generierten Gerätetyp. Folgende Typen sind dem Übersetzer bekannt (siehe auch „TMODE" im Manual Makroaufrufe [24]):

   ```
   8103    8122
   8150    8162
   8153    8160
   HOST    8124
   8151    8167
   8152    AP
   8110    9750
   8161    9003
   8121    9002
   PT80    9751
   T100    9752
   ```

Sollte ein Gerätetyp dem Übersetzer nicht bekannt sein, so werden die entsprechenden Zeichenpositionen mit Leerzeichen belegt.

6 Programmierhinweise

6.1 Programmierbare Testhilfen

Zum Testen von COBOL-Programmen stehen neben betriebssystemspezifischen Testhilfen (Dialogtesthilfe IDA) auch COBOL-Sprachelemente bereit. Es sind dies:

- Testhilfezeilen
  WITH DEBUGGING MODE-Klausel
- EXHIBIT-Anweisung
- TRACE-Anweisung

Testhilfezeilen sind COBOL-Standard. Die anderen Anweisungen sind Nicht-Standard.

6.1.1 Testhilfezeilen

Testhilfezeilen sind Zeilen im COBOL-Quellprogramm, die für Testzwecke verwendet werden sollen. Nach der Testphase können diese Zeilen ungeändert im Programm gelassen werden; es muß nur die WITH DEBUGGING MODE-Klausel aus dem SOURCE-COMPUTER-Paragraphen entfernt werden.

Ein D im Anzeigenbereich (Spalte 7) kennzeichnet eine Quellprogrammzeile als Testhilfezeile.

Bei Angabe der WITH DEBUGGING MODE-Klausel im SOURCE-COMPUTER-Paragraphen behandelt der Übersetzer Testhilfezeilen als normale Anweisungszeilen. Fehlt dagegen diese Klausel, so behandelt der Übersetzer Testhilfezeilen als Kommentarzeilen.

Der Inhalt einer Testhilfezeile muß so gewählt sein, daß bei der Übersetzung unabhängig von der Angabe der WITH DEBUGGING MODE-Klausel ein syntaktisch korrektes Programm entsteht. Testhilfezeilen sind im Quellprogramm erst nach dem OBJECT-COMPUTER-Paragraphen erlaubt.

Mehrere aufeinanderfolgende Testhilfezeilen sind zulässig. Die Fortsetzung von Testhilfezeilen ist gestattet; jedoch muß auch die Fortsetzungszeile ein D in Spalte 7 enthalten. Unzulässig ist die Trennung von Zeichenfolgen über mehrere Zeilen.

Format der WITH DEBUGGING MODE-Klausel:

WITH <u>DEBUGGING</u> <u>MODE</u>

Diese Klausel ist Teil des SOURCE-COMPUTER-Paragraphen.

6.1.2 EXHIBIT-Anweisung

Die EXHIBIT-Anweisung gibt aktuelle Werte von Datenfeldern an jeder beliebigen Stelle des Benutzerprogramms nach SYSLST aus.

Format:

```
        ⎧ NAMED          ⎫
EXHIBIT ⎨ CHANGED NAMED  ⎬ { bezeichner                  } ...
        ⎩ CHANGED        ⎭ { nichtnumerisches literal    }
```

Die Ausgabe ist bei den drei Varianten der EXHIBIT-Anweisung verschieden:

a) EXHIBIT NAMED . . .

   Bei jeder Ausführung einer EXHIBIT NAMED-Anweisung wird folgendes ausgegeben:

   1. die Namen der in der Anweisung angegebenen Bezeichner mit ihren aktuellen Werten;
   2. alle in der Anweisung angegebenen nichtnumerischen Literale.

b) EXHIBIT CHANGED NAMED . . .

   Bei der Ausführung einer EXHIBIT CHANGED NAMED-Anweisung wird folgendes ausgegeben:

   1. Wird die Anweisung zum ersten Mal ausgeführt, so wird der Name jedes in der Anweisung angegebenen Bezeichners sowie sein aktueller Wert ausgegeben. Bei jeder folgenden Ausführung werden Namen und Werte nur dann ausgegeben, falls sich inzwischen Werte verändert haben.
   2. Alle in der Anweisung angegebenen nichtnumerischen Literale.

c) EXHIBIT CHANGED . . .

   Bei der Ausführung einer EXHIBIT CHANGED-Anweisung wird folgendes ausgegeben:

   1. Wird die Anweisung zum ersten Mal ausgeführt, so wird der Name jedes in der Anweisung angegebenen Bezeichners sowie sein aktueller Wert ausgegeben. Bei jeder folgenden Ausführung werden nur inzwischen veränderte Werte ausgegeben.
   2. Alle in der Anweisung angegebenen nichtnumerischen Literale.

Regeln:

— Bezeichner darf kein Sonderregister sein, außer TALLY.

— Bezeichner mit variabler Länge sind nicht erlaubt.

— Die Ausgabe für jeden in einer EXHIBIT NAMED- oder EXHIBIT CHANGED NAMED-Anweisung angegebenen Bezeichner hat das Format:
  Leerzeichen
  Name des Bezeichners (einschließlich eventuell angegebener Kennzeichner)
  Leerzeichen
  Gleichheitszeichen
  Leerzeichen
  Wert des Bezeichners zur Ausführungszeit

— Ein Literal in der EXHIBIT-Anweisung wird bei der Ausgabe von Leerzeichen eingeschlossen.

— Jede Ausgabe von EXHIBIT-Anweisungen wird in festem, spaltengerechtem Format erzeugt.

— Numerische Werte haben Ausgabeformat, wie in der DISPLAY-Anweisung beschrieben (siehe Manual „COB1 Sprachbeschreibung", [1]).

— Enthalten zwei verschiedene EXHIBIT CHANGED- oder EXHIBIT CHANGED NAMED-Anweisungen gleiche Bezeichner, so verwenden sie verschiedene Bereiche für Wertesicherung (die Wertesicherung ist anweisungsbezogen). Abhängig vom Programmablauf können die zum Vergleich sichergestellten Werte eines Bezeichners oder mehrerer Bezeichner für die zwei Anweisungen verschieden sein.

- Bei zwei oder mehr Bezeichnern in einer EXHIBIT CHANGED- oder EXHIBIT CHANGED NAMED-Anweisung werden bei jeder Ausführung nur die geänderten Werte ausgegeben. Reservierte Stellen nicht ausgegebener Werte enthalten Leerzeichen.
- Die Länge für die Ausgabe aller Operanden darf die maximale Länge eines logischen Datensatzes von SYSLST nicht überschreiten.

Beispiel 34c: Verwendung der verschiedenen EXHIBIT-Anweisungen

Die Wirkung der drei verschiedenen EXHIBIT-Anweisungen wird gegenübergestellt. In allen Fällen lauten die Bezeichner I, J, K; nichtnumerisches Literal ist EINGABE-WERTE. Jede EXHIBIT-Anweisung wird dreimal ausgeführt. Den Wert von I ändert das Programm nicht; die Werte von J und K sind bei der dritten Ausführung verändert. Stets wird das nichtnumerische Literal ausgegeben.

```
EXHIBIT NAMED I,J,K,"EINGABE-WERTE"
EXHIBIT CHANGED NAMED I,J,K,"EINGABE-WERTE"
EXHIBIT CHANGED I,J,K,"EINGABE-WERTE"
```

Ausgabe der EXHIBIT NAMED-Anweisung:

```
1.)  I=000091 J=ABC K=95 EINGABE-WERTE
2.)  I=000091 J=ABC K=95 EINGABE-WERTE
3.)  I=000091 J=ABD K=96 EINGABE-WERTE
```

Ausgabe der EXHIBIT CHANGED NAMED-Anweisung:

```
1.)  I=000091 J=ABC K=95 EINGABE-WERTE
2.)                      EINGABE-WERTE
3.)           J=ABD K=96 EINGABE-WERTE
```

Ausgabe der EXHIBIT CHANGED-Anweisung:

```
1.)  000091 ABC 95 EINGABE-WERTE
2.)                EINGABE-WERTE
3.)         ABD 96 EINGABE-WERTE
```

6

Leerseite durch den Nachtrag vom August 1986

6.1.3 TRACE-Anweisung

Die Anweisung READY TRACE aktiviert die Testüberwachung.
Die Anweisung RESET TRACE beendet die Testüberwachung.
Testüberwachung heißt: Die Namen aller Kapitel und Paragraphen werden in der Reihenfolge ihrer Ausführung über SYSLST aufgelistet.

Format:

```
{READY} TRACE
{RESET}
```

Regeln:

- READY TRACE aktiviert die Testüberwachung für alle folgenden Kapitel und Paragraphen.
- RESET TRACE beendet die Wirkung einer vorangegangenen READY TRACE-Anweisung.
- Bei der Programmausführung mehrmals angesprochene Kapitel- bzw. Paragraphennamen werden auch mehrmals ausgegeben.

Leerseite durch den Nachtrag vom August 1986

6.2 Mehrfachbenutzbare Programme

6.2.1 Allgemeines

Häufig greifen gleichzeitig mehrere Prozesse auf ein gemeinsames Programm zu. Falls jeder Prozeß sein eigenes, vollständiges Programm lädt, kann das den Rechner stark belasten. Dies vermeidet man, indem man dieses Programm mehrfachbenutzbar (shareable) erklärt.

Mehrfachbenutzbare Programme werden wie folgt erzeugt:

- Der Anwender schreibt ein segmentierbares Programm.
- Der COB1-Übersetzer erzeugt "reentrant Code" (= wiederverwendbaren Code) aus allen Anweisungen der Procedure Division (LC7) in verschiedenen Segmenten.
- Der Systemverwalter macht mit dem SHARE-Kommando bestimmte Segmente (Programm-Moduln) aus LC7 mehrfachbenutzbar. Alle übrigen LC-Abschnitte (LC0 bis LC6 und LC8) sind nicht mehrfachbenutzbar.

6.2.2 Segmentierung

Zweck der Segmentierung ist es, ein Programm in Teile — sog. Segmente — zu zerlegen.

Ein segmentiertes Programm (siehe [1]) besteht aus einem speicherresidenten Grundsegment (auch Wurzelsegment) und einem oder mehreren überlagerbaren Segmenten. Dies sind entweder feste oder unabhängige Überlagerungssegmente. Nur aus unabhängigen Überlagerungssegmenten können mehrfachbenutzbare Moduln gemacht werden.

Ein Programm wird segmentiert durch

- Einteilung in Kapitel (Sections), die Section-Nummern tragen (0 . . . 99)
- die SEGMENT-LIMIT-Klausel (1 . . . 49).

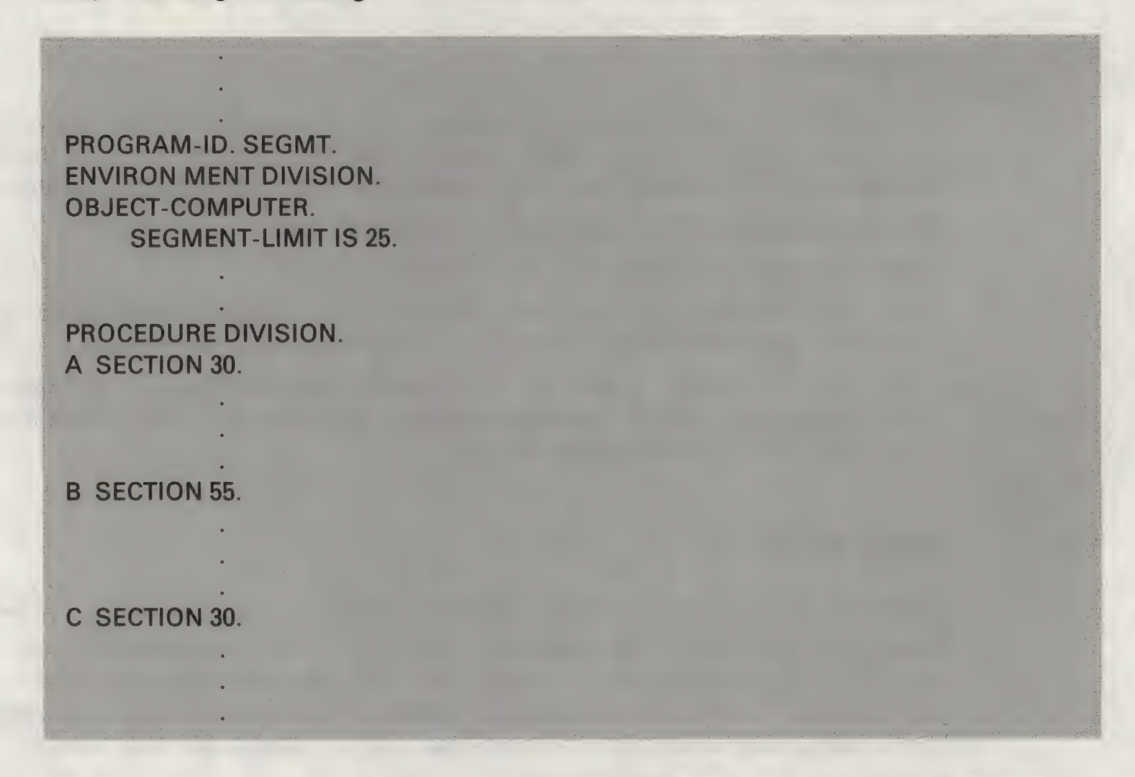
Die SEGMENT-LIMIT-Klausel bestimmt, welche Programmteile zum Grundsegment gehören und welche Programmteile feste Überlagerungssegmente sein sollen (Näheres siehe [1]). Aus Kapiteln mit Section-Nummern <Angabe in der SEGMENT-LIMIT-Klausel wird genau ein Grundsegment gebildet. Kapitel mit Section-Nummern zwischen SEGMENT-LIMIT und 49 definieren feste Überlagerungssegmente. Fehlt die SEGMENT-LIMIT-Klausel, so gehören alle Kapitel mit Nummern <50 zum Grundsegment. Hingegen bestimmen Kapitel mit Section-Nummern ≧50 unabhängige Überlagerungssegmente. Jedes Überlagerungssegment wird aus Kapiteln gleicher Section-Nummer erzeugt.

Aus je einem Segment erzeugt der COB1-Übersetzer genau einen Bindemodul, der in die temporäre Bindemoduldatei * eingetragen wird. Das Grundsegment bildet dabei den Hauptmodul.

Der Name des Hauptmoduls leitet sich vom Programmnamen aus der PROGRAM-ID ab. Daraus ergeben sich die Namen aller übrigen Moduln durch Anfügen der jeweils zugehörigen Section-Nummer (im folgenden mit n_1n_2 bezeichnet).

Hauptmodulname	Modulname
X	X n_1n_2
XX	XX n_1n_2
XXX	XXX n_1n_2
XXXX	XXXX n_1n_2
XXXXX	XXXXX n_1n_2
XXXXXX bis XXXXXXY	XXXXXX n_1n_2

Beispiel 35: Segmentierung



```
          .
          .
          .
PROGRAM-ID. SEGMT.
ENVIRON MENT DIVISION.
OBJECT-COMPUTER.
      SEGMENT-LIMIT IS 25.
          .
          .
PROCEDURE DIVISION.
A  SECTION 30.
          .
          .
          .
B  SECTION 55.
          .
          .
          .
C  SECTION 30.
          .
          .
          .
          .
```

Daraus erzeugt COB1 zusätzlich zum Hauptmodul SEGMT die Moduln SEGMT30 und SEGMT55:

A SECTION 30 C SECTION 30	⟶	Segment 30 (fest, überlagerbar)	⟶	SEGMT30
B SECTION 55	⟶	Segment 55 (unabhängig, überlagerbar)	⟶	SEGMT55

Da die letzten 2 Zeichen immer n_1n_2 enthalten, ist vom Anwender dafür Sorge zu tragen, daß keine mehrfachen Namensdefinitionen auftreten.

Beispiel 36: Auszüge aus einem segmentierten Programm

```
ACOB1.30   COBOL-74 COMPILATION                    SOURCE LISTING
             V      VV   V
        1            IDENTIFICATION DIVISION.
        2            PROGRAM-ID.                 STUDY.
        3            AUTHOR.                     RUDOLF.
        4           *
        5            ENVIRONMENT DIVISION.
        6            CONFIGURATION SECTION.
        7            OBJECT-COMPUTER.
        8                SEGMENT-LIMIT IS 20.
                              :
       66            PROCEDURE DIVISION.
       67            EINLESEN SECTION 30.
                              :
       83            BESTAND SECTION 50.
                              :
      109            SORTIEREN SECTION 80.
      110            BE.
      111                OPEN OUTPUT AUSGABE.
      112            SORTIER.
      113                SORT SORTARB ON ASCENDING VER , TIT
      114                         INPUT PROCEDURE IS ISAM-SAM
      115                         OUTPUT PROCEDURE IS SAM-SAM.
      116           *
      117            ISAM-SAM SECTION 80.
      118            BEG.
      119                OPEN INPUT BESTAND.
      120            ISAM.
      121                READ BESTAND NEXT RECORD AT END GO TO ISEND.
      122                MOVE B-NUMMER TO BES.
      123                MOVE B-VER TO VER.
      124                MOVE B-TIT TO TIT.
      125                RELEASE SO-SATZ.
      126                GO TO ISAM.
      127            ISEND.
      128                CLOSE BESTAND.
      129           *
      130            SAM-SAM SECTION 80.
```

Hinweis:

Die erzeugten Objektmoduln kann man anhand der Protokolle eines LMR-Laufes (vgl. Beispiel 26b, S. 3-1) und eines Binder-Laufes (vgl. Beispiel 28, S. 4-9) kontrollieren.

6.2.3 Shared Code

Segmente mit Section-Nummer >49 können mehrfachbenutzbar gemacht werden. Der Anwender geht dabei so vor:

- Die nach obigem Schema vergebenen Modulnamen gestatten, daß jene Moduln, die mehrfachbenutzbar gemacht werden sollen, in einer Bindemodulbibliothek mit Hilfe von LMR [3] oder LMS [21] abgelegt werden.
- Diese Moduln erklärt der Systemverwalter als "shareable" (mit dem SHARE-Kommando).
- Sie werden mit folgenden Kommandos aufgerufen:

  falls der vorgebundene Objektmodul in der Bibliothek steht:

  /EXEC (modulname,bibliotheksname)

  b) falls der vorgebundene Objektmodul in der *-Datei steht und mehrfachbenutzbare Moduln von der Modulbibliothek hinzugebunden werden:

  /SYSFILE TASKLIB = dateiname
  /EXEC *

Shared-Code-Segmente — also mehrfachbenutzbare Überlagerungssegmente — werden geladen, sobald der erste Prozeß in einer BS2000-Sitzung sie anfordert. Sie bleiben dann bis zum Ende der Sitzung im Klasse-4-Speicher des Systems für alle Prozesse verfügbar. Alle nicht mehrfachbenutzbaren Teile eines Programms werden pro Prozeß bzw. pro Anwender in den Klasse-6-Speicher geladen.

Das folgende Bild veranschaulicht Programmläufe ohne bzw. mit Shared Code.

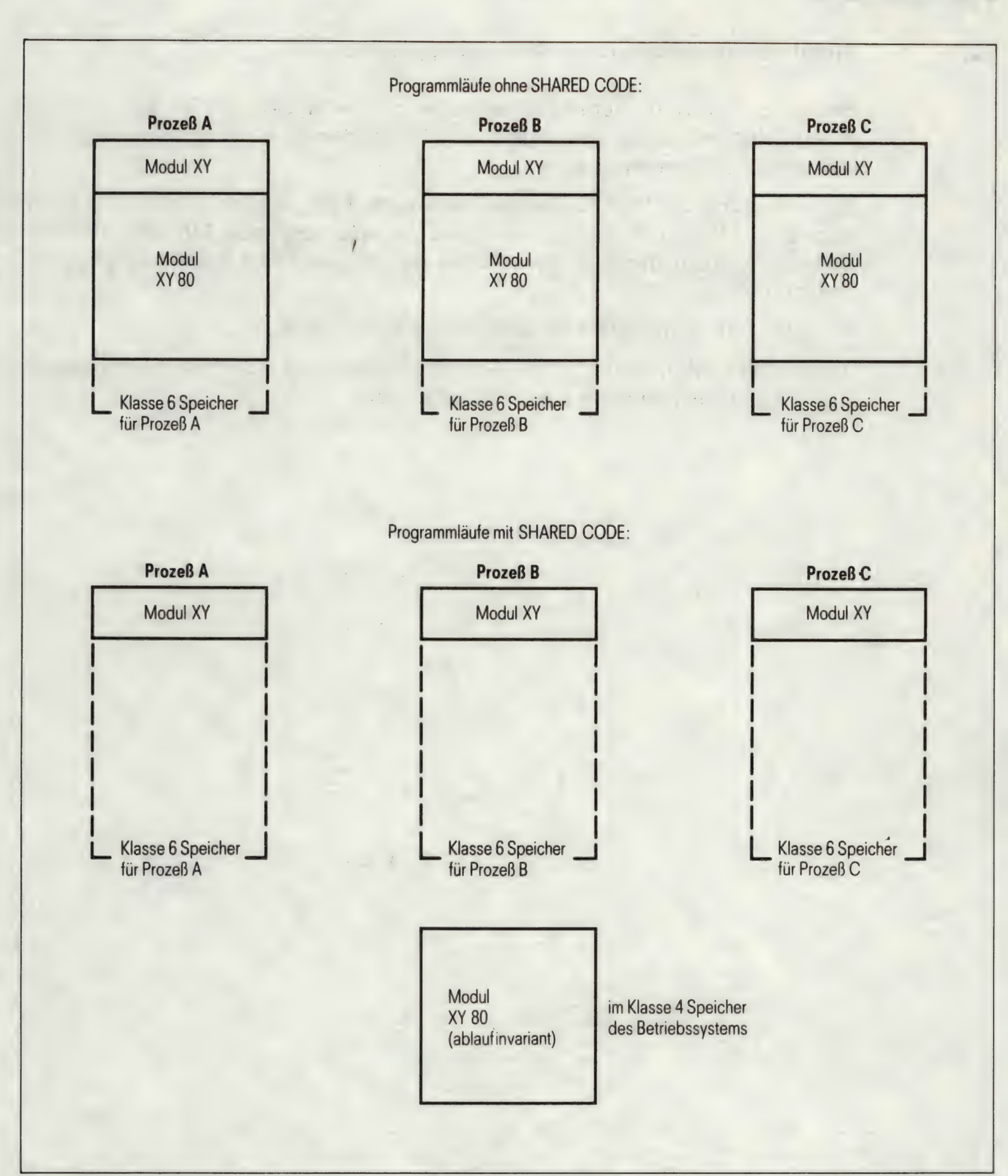


Bild 6-1

Shared Code
Oben: Der Modul XY 80 wird dreimal geladen (Klasse-6-Speicher).
Unten: Der Modul XY 80 wird nur einmal geladen (Klasse-4-Speicher).

6.2.4 Mehrfachbenutzbarkeit des Ablaufzeitsystems

Das COB1-Laufzeitsystem kann teilweise mehrfachbenutzbar in den Arbeitsspeicher geladen werden und damit von allen parallellaufenden COB1-Programmen gleichzeitig benutzt werden (Speicherersparnis).

Zur Entlastung der SHARE-Tabelle (siehe Dienstprogramm DLL) werden alle mehrfach-benutzbaren Moduln in einem Großmodul zusammengefaßt. Mit dem SHARE- und LOAD-Kommando kann der Systemverwalter diesen und einen Verbindungsmodul in Klasse-4-Speicher laden.

Der Ablauf der Programme muß mit Hilfe des DLL erfolgen.

Die normalen Module des COB1-Ablaufzeitsystems und die mehrfach-benutzbaren Moduln dürfen nicht zusammen in einer Bibliothek sein.

6.3 Programmverknüpfungen

Bei der Erstellung von Programmsystemen ist es oftmals zweckmäßig, die Komponenten in verschiedenen, auf die jeweiligen Teilprobleme zugeschnittenen Programmiersprachen zu schreiben.

Zu den erforderlichen Programmverknüpfungen dienen Verfahren und Konventionen, die im folgenden beschrieben werden.

Übersicht:

Es sind folgende Programmverknüpfungen beschrieben:

1. COBOL-COBOL
2. COBOL-Assembler (Assembler-COBOL)

Hinweis: Zur Verwendung von FOR1-Programmen als COBOL-Unterprogramme siehe „FOR1 Benutzerhandbuch" [16].

6.3.1 Programmverknüpfungen COBOL-COBOL

(Siehe auch Kapitel Programmkommunikation in der COB1-Beschreibung [1]).

Der Aufruf eines Unterprogramms erfolgt gemäß ANS 74 COBOL-Standard durch folgende Anweisung:

```
CALL literal [USING name . . .]
```

Wobei literal der „Unterprogrammname" ist und name entweder Datenname, Dateiname oder Prozedurname bedeutet.

Der Unterprogrammname muß mit einem Alphazeichen beginnen, darf nur Buchstaben und Ziffern enthalten und maximal 8 Zeichen lang sein.

Die USING-Liste darf maximal 216 Operanden enthalten.

Ist name ein Datenname, wird dem Unterprogramm dessen Adresse zur Verfügung gestellt.

Jeder in der USING-Liste des aufrufenden Programms angegebene Datenname muß in der FILE SECTION, der WORKING-STORAGE SECTION oder der LINKAGE SECTION definiert sein. Er kann jede Stufennummer außer 88 haben.

Ist name ein Dateiname, wird dem Unterprogramm die Adresse des zugehörigen FCB zur Verfügung gestellt.

Ist name ein Prozedurname (Kapitel oder Paragraph), wird die Anfangsadresse dieser Prozedur übergeben.

Ein aufgerufenes Programm kann CALL-Anweisungen enthalten, jedoch keine, die direkt oder indirekt das aufrufende Programm aufrufen.

Es wird noch folgendes Aufrufformat unterstützt:

```
ENTER LINKAGE.
CALL unterprogrammname [USING name...]
ENTER COBOL.
```

Einsprung

Der Einsprung in ein Unterprogramm erfolgt gemäß ANS'74 COBOL-Standard durch folgende Anweisung:

PROCEDURE DIVISION [USING data-name . . .].
wobei data-name gleich datenname ist.

Die USING-Liste des Einsprungs muß gleichviele Operanden wie die des Aufrufs enthalten.

Jeder in der USING-Liste des aufgerufenen Programms angegebene Datenname muß in der LINKAGE SECTION definiert sein und darf nur die Stufennummern 01 oder 77 haben.

Die Zuordnung der übergebenen Operanden erfolgt über ihre Stellung in der Operandenliste.

Darüberhinaus können weitere Einsprungpunkte wie folgt definiert werden (Erweiterung des ANS'74 bzw. ANS'68 COBOL-Standards):

1. ENTRY einsprungname [USING data-name . . .].
2. ENTER LINKAGE.
   ENTRY einsprungname [USING data-name . . .].
   ENTER COBOL.

Diese Anweisungen können an beliebiger Stelle innerhalb der PROCEDURE DIVISION aufgeführt werden.

Bei diesen Formaten ist zu beachten, daß der Unterprogrammname vom Einsprungnamen verschieden sein muß und beim Aufruf in der CALL-Anweisung der Einsprungname zu verwenden ist.

Rücksprung

Der Rücksprung erfolgt gemäß ANS'74 COBOL-Standard durch folgende Anweisung:

EXIT PROGRAM.

Diese Anweisung muß die einzige in einem Programmsatz sein. Dieser Programmsatz muß der einzige in einem Paragraphen sein.

Unterstützt wird noch folgendes Format (DOD-COBOL bzw. ANS'68 COBOL-Standard):

ENTER LINKAGE.
RETURN.
ENTER COBOL.

Datenkonventionen

Im aufgerufenen Programm werden die in der USING-Liste angegebenen Operanden entsprechend ihren in der LINKAGE SECTION angegebenen Datenbeschreibungen behandelt.

Hinweis:
Es werden nur die Adressen der Daten übergeben (call by reference), jedoch keine Angaben über deren Format und Ausrichtung. Für die notwendige und richtige Ausrichtung, sowie die formatgerechte Verarbeitung der übergebenen Daten ist der Programmierer selbst verantwortlich.

Konventionen zur Unterprogrammtechnik

Bei der Verknüpfung von COBOL-Programmen gelten folgende Konventionen:

1. Das aufrufende Programm stellt einen Sicherstellungsbereich zur Verfügung.
2. Das aufrufende Programm versorgt die Parameteradreßliste und die Mehrzweckregister entsprechend den Registerkonventionen.
3. Das aufgerufene Programm sichert alle Mehrzweckregister mit Ausnahme von Register 13 im Sicherstellungsbereich des aufrufenden Programms.
4. Das aufgerufene Programm führt die Vorwärts- und Rückwärtsverkettung der Sicherstellungsbereiche durch.
5. Vor dem Rücksprung lädt das aufgerufene Programm die Mehrzweckregister aus dem Sicherstellungsbereich des aufrufenden Programms.

6. Das COBOL-Sonderregister RETURN-CODE dient zur Verständigung zwischen getrennt übersetzten COBOL-Moduln, die zum Ablauf in einem ladefähigen Programm zusammengebunden wurden. Das Sonderregister existiert nur einmal im Programm und ist intern als vierstelliges Datenfeld (PIC S9(4) COMP SYNC VALUE ZERO) definiert. RETURN-CODE kann während des Ablaufs beliebig von einzelnen Moduln abgefragt oder verändert werden. Bei Beendigung des Programmlaufs wird vom Laufzeitsystem überprüft, ob RETURN-CODE auf Null steht. Ist das nicht der Fall, wird die Fehlermeldung 9040 ausgegeben. Wurde das Programm innerhalb einer Prozedur aufgerufen, verzweigt das System zum nächsten STEP-, ABEND-, ABORT-, ENDP- oder LOGOFF-Kommando.

Registerkonventionen

Die nachfolgende Tabelle gibt eine Übersicht über die Registerversorgung, die das aufrufende Programm vor dem Ansprung des aufgerufenen Programms durchführt.

Tabelle: Registerbenutzung

Register nummer	Registerverwendung	Inhalt
0	Arbeitsregister	Dient bei Verknüpfung von COBOL-Moduln als Register zur Koordinierung
1	Parameterübergabe	Adresse der Parameteradreßliste
13	Sicherstellungsbereich	Adresse des Sicherstellungsbereiches des aufrufenden Programms
14	Rücksprung	Adresse des Rückkehrpunktes ins aufrufende Programm.
15	Einsprungpunkt	Adresse des Einsprungpunktes im aufgerufenen Programm.

Aufbau des Sicherstellungsbereiches

Die Struktur des Sicherstellungsbereiches im aufrufenden Programm ist wie folgt:

Wort	Inhalt
1	wird von COBOL intern verwendet
2	enthält die Anfangsadresse des Sicherstellungsbereiches des aufrufenden Programms. Im ersten **aufrufenden** Programm ist der Inhalt dieses Feldes 0.
3	enthält die Anfangsadresse des Sicherstellungsbereiches des aufgerufenen Programms. Im letzten **aufgerufenen** Programm ist der Inhalt dieses Feldes nicht verwertbar.
4	Register 14
5	Register 15
6	Register 0
...	....
18	Register 12

Aufbau der Operandenadreßliste

Die Operandenadreßliste besteht aus Adreßkonstanten. Diese enthalten die Adressen der in der Operandenliste aufgeführten Operanden. Die Adreßkonstanten stehen in derselben Reihenfolge, wie die Operanden in der Operandenliste. Die letzte Adreßkonstante wird durch eine 1 im höchstwertigen Bit gekennzeichnet.

6.3.2 Programmverknüpfung COBOL-ASSEMBLER (ASSEMBLER-COBOL)

Beim Aufruf von Unterprogrammen, die in Assembler geschrieben sind, oder beim Aufruf von COBOL-Unterprogrammen aus Assemblerhauptprogrammen, sind bestimmte Konventionen zu beachten. Die zur Verknüpfung notwendigen Sicherstellungsbereiche, ihre Lage und Adressierungstechnik werden hier näher erläutert. Außerdem wird die Übergabe bzw. Übernahme von Operanden mitsamt den zugehörigen Registerkonventionen beschrieben.

Allgemeines

Ein COBOL-Programm, das CALL- oder ENTRY-Anweisungen enthält oder vom Übersetzer als Unterprogramm erkannt wird, veranlaßt automatisch die Generierung der notwendigen Befehlsfolgen und Sicherstellungsbereiche zur Verknüpfung dieses Programmes mit anderen Programmen. Programme, die in Assembler geschrieben sind, müssen den nachfolgend beschriebenen Verknüpfungskonventionen genügen, damit sie in Verbindung mit COBOL-Programmen, als rufende oder aufgerufene Programme, verwendet werden können.

Die folgende Tabelle zeigt die Benutzung der allgemeinen Register im Zusammenhang mit Programmverknüpfungen.

Tabelle: Registerbenutzung

Register-Nummer	Registerverwendung	Inhalt
1	Operandenübergabe	Adresse der Operandenadressen, die an das aufgerufene Programm übergeben werden sollen.
13	Sicherstellungs-bereich	Adresse eines 18 Worte (= 72 Byte) langen Bereiches im laufenfenden Programm, der vom aufgerufenen Programm zur Sicherstellung der Register verwendet werden kann.
14	Rücksprung	Adresse des Rückkehrpunktes ins aufrufende Programm.
15	Einsprungpunkt	Adresse des Einsprungpunktes im aufgerufenen Programm.

Konventionen in einem aufgerufenen Assembler-Programm

Ein aufgerufenes Assembler-Unterprogramm muß den Inhalt der von ihm benutzten Register und anderer in der Tabelle angegebener Informationen im Sicherstellungsbereich abspeichern, dessen Adresse ihm in Register 13 beim Aufruf übergeben wurde. Ein aufgerufenes Programm muß außerdem eine Rücksprungroutine besitzen, deren Aufgabe es ist,

1. die Adresse des Sicherstellungsbereiches des aufrufenden Programmes in Register 13 zurückzuladen,
2. den Inhalt anderer benutzter Register wiederherzustellen,
3. die Rückkehradresse in Register 14 zu laden und schließlich,
4. in das aufrufende Programm an die Adresse, die in Register 14 enthalten ist, zurückzuspringen.

Der COB1-Übersetzer setzt die Programm-Maske auf X'00'. Deshalb führen im Assembler-Unterprogramm die folgenden Fehler zu keiner Programmunterbrechung:

IW = X'6C' Mantisse = 0
IW = X'70' Exponentialunterlauf
IW = X'74' Dezimalüberlauf
IW = X'78' Festpunktüberlauf

Falls der Programmierer die Programm-Maske im Assembler-Unterprogramm verändert, muß er sie vor dem Rücksprung in das COBOL-Programm auf X'00' zurücksetzen.

Tabelle: Inhalt und Aufbau des Registersicherstellungsbereiches

Wort	Adresse	Inhalt
1	Bereich	Länge des Sicherstellungsbereiches
2	Bereich + 4	Adresse (eingetragen durch das aufgerufene Programm) des vom aufrufenden Programm verwendeten Sicherstellungsbereiches. Es handelt sich also um die Adresse des Sicherstellungsbereiches, die an das aufrufende Programm übergeben wurde, falls dieses ebenfalls ein aufgerufenes Programm war. (Rückwärtskettungsadresse).
3	Bereich + 8	Adresse (eingetragen vom aufgerufenen Programm) des nächsten Sicherstellungsbereiches, d. h. des Sicherstellungsbereiches des aufgerufenen Programmes, den dieses für ein von ihm aufgerufenes Programm zur Verfügung stellt. (Dieser Sicherstellungsbereich ist nicht erforderlich, falls kein weiterer Unterprogrammaufruf erfolgen soll.)
4	Bereich + 12	Rückkehradresse (Inhalt von Register 14), eingetragen vom aufgerufenen Programm.
5	Bereich + 16	Einsprungadresse (Inhalt von Register 15), eingetragen vom aufgerufenen Programm.
6	Bereich + 2Ø	Inhalt von Register Ø, eingetragen vom aufgerufenen Programm.
7	Bereich + 24	Inhalt von Register 1 (eingetragen vom aufgerufenen Programm). Register 1 enthält die Adresse der Parameterliste, die an das aufgerufene Programm übergeben wurde.
8 . . . 18	Bereich + 28 . . . Bereich + 68	Inhalt der Register 2 bis 12, eingetragen durch das aufgerufene Programm.

Beispiel 37: Aufgerufenes Assembler-Programm

```
UPRO          START
              .
              .
              .
              ENTRY ASSUPRO
*
*
* Definition des Sicherstellungsbe-
* reiches als DUMMY SECTION
SAVAREA       DSECT
BEREICH       DS        F
RUECK         DS        F
VORWAERT      DS        F
REG14         DS        F
REG15         DS        F
REGØ          DS        F
REG1          DS        F
REG2TO12      DS        11F
UPRO          CSECT
              USING     SAVAREA,13
```

Definition eines Einsprungpunktes mit Namen ASSUPRO, auf den ein anderes Programm Bezug nehmen kann.

Code	Erläuterung
`ASSUPRO   STM   14, 12, REG14`	Register 14, 15, Ø und 1 werden in den dafür vorgesehenen Abschnitt REG14, REG15, REGØ bis REG1 des Sicherungsbereiches abgespeichert.
`*` `*` `*`	Mit dem gleichen Befehl werden auch die nicht für die Programmverknüpfung reservierten Register 2 bis 12 im Sicherstellungsbereich (REG2TO12) abgesetzt.
`*` `*`	Die Sicherstellung von Register 2 bis 12 hängt von deren Benutzung durch das aufgerufene Programm ab und kann unterbleiben für nicht verwendete Register.

`          LR    5, 13`
`*`
`*`
	Die Adresse des Sicherstellungsbereiches des aufrufenden Programmes wird in Register 5 zwischengespeichert und kann zur Adressierung des Sicherungsbereiches verwendet werden.

`          L     13, SAVADDR` `          DROP  13`	Die Adresse des eigenen Sicherstellungsbereiches wird in Register 13 geladen.

`          USING SAVAREA, 5`
`          ST    13, VORWAERT`
`*`
	Die Adresse des eigenen Sicherstellungsbereiches wird in das Feld VORWAERT (Wort 3) des Sicherstellungsbereiches des aufrufenden Programmes abgespeichert.

`          USING SAVAREA, 13`
`          ST    5, RUECK`
`*`
	Die Adresse des Sicherstellungsbereiches des aufrufenden Programmes wird in das Feld RUECK (Wort 2) des eigenen Sicherstellungsbereiches eingetragen.

`          .`
`          .`
`          .`
`          .`
`     Benutzeranweisungen`
`          .`
`          .`
`          .`
`SAVADDR   DC    A(SAVAREAI)`
`          .`
`          .`
`          .`
`SAVAREAI  DC    18F'Ø'`
`          .`
`          .`
`          .`
`* Rücksprungroutine`
`          USING SAVAREA, 13`	

`          L     13, RUECK`
`*`
`*`
	Die Adresse des Sicherstellungsbereiches des aufrufenden Programmes wird aus den eigenen Sicherstellungsbereich in Register 13 zurückgeladen.

`          LM    14, 12, REG14`
`*`
`*`
	Die Inhalte der Register 14, 15, Ø, 1 und 2 bis 12 des aufrufenden Programmes werden wiederhergestellt.

`          BR    14`	Rücksprung ins aufrufende Programm

Konventionen in einem aufrufenden Assembler-Programm

Ein aufrufendes Assembler-Programm muß einen Sicherstellungsbereich der Länge 18 Worte (= 72 Bytes), der an einer Wortgrenze beginnt, zur Verfügung stellen. Dieser Bereich wird vom aufgerufenen Programm zum Abspeichern der Registerinhalte des aufrufenden Programms verwendet. Die Adresse dieses Sicherstellungsbereiches muß in Register 13 an das aufgerufene Programm übergeben werden. Falls das aufrufende Programm Operanden übergeben soll, muß die Adresse der Operandenliste in Register 1 übergeben werden. Ferner muß das aufrufende Programm die Adresse des Rückkehrpunktes in Register 14 übergeben, die Adresse des Einsprungpunktes des aufgerufenen Programmes muß in Register 15 enthalten sein.

Die Operandenliste ist eine Gruppe aufeinanderfolgender Worte, deren Inhalt die Adressen von Datenfeldern darstellt, die an das aufgerufene Programm übergeben werden sollen. Diese Operandenliste muß auf Wortgrenze beginnen. Das höchstwertige Bit des letzten Eintrags dieser Adreßliste wird auf 1 gesetzt, um das Ende der Operandenliste anzugeben.

Beispiel 38: Aufrufendes Assembler-Programm

```
         .
.        .
.        .
         L      13,SAVADDR
*
         .
         .
         .
SAVADDR  DC     A(BEREICH)
         .
         .
         .
BEREICH  DC     18F'Ø'
         .
         .
         .
* Unterprogrammaufruf
         L      1,ADRPARAM
*
         L      15,ENTRADDR
*

         BALR   14,15
*


         .
         .
         .
* Adreßkonstanten für Unterpro-
* grammeinsprungpunkte
         EXTRN  UPROENT
ENTRADDR DC     A(UPROENT)
* Operandenliste und Adreßkonstante
```

- L 13,SAVADDR: Die Adresse des Sicherstellungsbereiches wird in Register 13 geladen.
- L 1,ADRPARAM: Register 1 wird mit der Adresse der Parameteradreßliste geladen.
- L 15,ENTRADDR: Register 15 wird mit der Adresse des Einsprungpunktes des aufzurufenden Programmes geladen.
- BALR 14,15: Das aufzurufende Programm wird angesprungen und die Rückkehradresse in Register 14 übergeben.
- EXTRN UPROENT / ENTRADDR DC A(UPROENT): oder ENTRADDR DC V(UPROENT)

```
ADRPARAM DC     A(PARAMLST)          Adresse der Operandenliste
         .
         .
         .
         DS     ØF
PARAMLST DC     A(PARAM1)            Adresse des ersten Operanden
         DC     A(PARAM2)            Adresse des zweiten Operanden
         DC     X'8Ø'                Indikator für letzte Operandenadresse
         DC     AL3(PARAM3)          Adresse des dritten (letzten) Operanden
         .
         .
         .
* Operandenbereiche
         .
         .
         .
PARAM1   DC     C'ERSTER OPERAND'
         .
         .
         .
PARAM2   DC     C'ZWEITER OPERAND'
         .
         .
         .
PARAM3   DC     C'LETZTER OPERAND'
         .
         .
         .
```

Bemerkung:
Falls das aufrufende Programm bereits von einem anderen Programm gerufen wurde, würde es zusätzliche Anweisungen enthalten (vgl. Beispiel 37).

6.4 Ein-Ausgabe von Daten über Systemdateien

Folgende COBOL-Anweisungen (siehe auch Manual „COB1-Sprachbeschreibung") greifen auf Systemdateien zu (vgl. Tabelle unten):

a) EXHIBIT
b) TRACE (die durch READY TRACE eingeschalteten und durch RESET TRACE ausgeschalteten Testüberwachungsausgaben)
c) STOP literal
d) ACCEPT
e) DISPLAY

Zu a) und b): Die Ausgabe erfolgt über SYSLST (Beschreibung dieser Anweisungen im Abschnitt 6.1).

Zu c): Diese Anweisung unterbricht das laufende Programm mit einer Meldung am Bedienungsplatz. Der Operateur setzt das Programm mit einer beliebigen, programmunabhängigen Eingabe fort.

Höchstens 122 Bytes vom angegebenen Literal werden auf dem Bedienungsplatz ausgegeben. — Eine explizite Umweisung auf ein anderes Ausgabegerät ist nicht möglich.

Der Anwender bekommt die Meldung 9080 über SYSOUT, wenn die Anweisung STOP literal erreicht ist.

Zu d) und e): Auf zweierlei Art verwendet man Systemdateien:

- direkte Angabe:
  Bei ACCEPT sind möglich SYSIPT (Standard), SYSRDR, SYSIN, TERMINAL, CONSOLE.
  Bei DISPLAY sind möglich SYSLST (Standard), SYSOPT, SYSP[UN]CH, SYSOUT, TERMINAL, CONSOLE.
- über Merknamen:
  ACCEPT. . . FROM merkname bzw.
  DISPLAY. . . UPON merkname.

Die Verknüpfung von Systemdatei und Merkname definiert man im SPECIAL-NAMES-Paragraphen der ENVIRONMENT DIVISION.

Satzformate und Satzlängen

Die einzelnen Systemdateien bzw. Geräte verarbeiten unterschiedliche Satzlängen. Sie verwenden entweder Satzformat F oder Satzformat V; näheres zeigt die folgende Tabelle.

Tabelle: COBOL-Anweisungen, Systemdateien, Satzformate und Satzlängen

COBOL-Anweisungen	zugehörige Systemdateien des BS2000/Gerät	Satzformat	logische Satzlänge der Systemdatei
ACCEPT...FROM TERMINAL ACCEPT...FROM SYSRDR ACCEPT...FROM SYSIN	SYSDTA	F	bei Zuweisung auf Kartenleser max. 80 Bytes
		V	bei Eingabe über Datenstation oder von Plattendatei: max. 32 K
ACCEPT... ACCEPT...FROM SYSIPT	SYSIPT	F	80 Bytes
ACCEPT...FROM CONSOLE[1])	Bedienungsplatz	V	max. 72 Bytes
DISPLAY...UPON CONSOLE[1]) STOP literal[2])	Bedienungsplatz	V	max. 127 Bytes
DISPLAY...UPON TERMINAL DISPLAY...UPON SYSOUT	SYSOUT	V	im Stapelbetrieb: max. 2039 Bytes
			im Dialogbetrieb: max. 32 K

COBOL-Anweisungen	zugehörige Systemdateien des BS2000/Gerät	Satz-format	logische Satzlänge der Systemdatei
DISPLAY... DISPLAY...UPON SYSLST EXHIBIT... READY TRACE RESET TRACE	SYSLST	V	max. 133 Bytes: 1 Byte Steuerinformation, 132 Bytes Daten
DISPLAY...UPON SYSOPT DISPLAY...UPON SYSPUNCH	SYSOPT	F	max. 80 Bytes: 72 Datenbytes; Bytes 73—80 enthalten die ersten 8 Bytes des PROG.-ID-Namens.

[1]) CONSOLE sollte nur in Ausnahmefällen verwendet werden, da im „Closed Shop"-Betrieb des BS2000 der Operateur meist nicht über die verlangten Daten Bescheid weiß. Statt dessen kann man TERMINAL einsetzen.

[2]) Das Literal darf höchstens 122 Bytes lang sein.

Übertragen werden Daten in der Länge des im COBOL-Programm beschriebenen Feldes.

Falls der Anwender dieses Feld nicht vollständig füllt,

- wird bei ACCEPT dieses Feld mit Leerzeichen gefüllt
- besteht bei DISPLAY der Rest aus altem Feldinhalt, ggf. aus binären Nullen (nicht abdruckbar).

Falls Daten über ein ACCEPT-Feld hinausgehen, werden sie nicht übertragen.

Zu beachten ist, falls das Satzformat F verwendet wird (abhängig von der Systemdatei bzw. dem zugewiesenen Gerät): Ist die Länge eines Bezeichners in einer ACCEPT-Anweisung größer als die logische Satzlänge der Systemdatei, so werden Daten nachgefordert, d. h. mehrere Eingabeoperationen (Makroaufrufe) veranlaßt.

Bei abnormaler Beendigung eines Einlesevorganges durch Erreichen von EOF der Eingabedatei wird die Meldung 9051 oder 9052 über SYSOUT ausgegeben. Es wird /* in den Eingabebereich übertragen und mit der folgenden Anweisung fortgefahren.

Beispiele:

38a) Ein Programm mit ACCEPT- und DISPLAY-Anweisungen ohne Verwendung von Merknamen ist im Beispiel 1 abgedruckt.

38b) Ein vom Benutzer gewählter Merkname wird im SPECIAL-NAMES-Paragraphen der ENVIRONMENT DIVISION mit der Systemdatei SYSIPT verknüpft:

```
       .
       .
       .
ENVIRONMENT DIVISION.
       .
       .
       .
SPECIAL-NAMES.
    SYSIPT IS EINGABE.
       .
       .
       .
PROCEDURE DIVISION.
       .
       .
       .
    ACCEPT DATEN FROM EINGABE.
       .
       .
       .
```

6.5 Dateibearbeitung

6.5.1 Allgemeines

Dieser Abschnitt behandelt katalogisierte Dateien. Ausgenommen sind Systemdateien sowie COBOL-Bibliotheken.

Um logische Sätze zu speichern und wieder aufzufinden, bedient sich das COB1-System bestimmter Zugriffsmethoden des Datenverwaltungssystems (DVS), nämlich SAM, ISAM, UPAM (siehe Manual „Datenverwaltungssystem" [4]).

Der Name einer vom DVS verwalteten Datei ist (neben anderen Dateieigenschaften) im Systemkatalog eingetragen. Auf ihn beziehen sich etwa die Dateinamen im FILE-Kommando oder im CATALOG-Kommando. Der Name einer katalogisierten Datei ist nicht notwendigerweise identisch mit dem (logischen) Dateinamen im COBOL-Programm. Dort ist eine Datei durch den Dateinamen (maximal 30 Zeichen; davon sind nur die ersten 8 Zeichen signifikant) in der Dateierklärung (FD) der FILE SECTION definiert. Dieser logische Dateiname muß zur Ablaufzeit mit einer physischen Datei verbunden werden, und zwar durch das FILE-Kommando (siehe Abschnitt 5.1.3):

/FILE dateiname,LINK=linkname

Für den Zugriff zu Datensätzen hält COBOL bestimmte Sprachmittel bereit (Genaueres siehe Manual „COB1 Sprachbeschreibung" [1]):

Tabelle: Zugriff zu Datensätzen

COBOL-Sprachelement	Bedeutung
ORGANIZATION-Klausel	Organisationsform
ACCESS-Klausel	Zugriffsart
OPEN-Anweisung	Dateieröffnung

6.5.2 Dateiorganisationsformen, Zugriffsarten, Dateieröffnung, Übertragung der Daten

Organisationsformen

Eine von einem COBOL-Programm zu verarbeitende Datei kann eine der folgenden Organisationsformen haben:

- sequentiell (DVS: SAM)
- relativ (DVS: UPAM)
- indiziert (DVS: ISAM)

Die vom Benutzer gewählte Organisationsform bestimmt die Zugriffsmethode des DVS, die in Klammer jeweils angegeben ist.

Sequentiell organisierte Dateien können auf jedem möglichen Ein- oder Ausgabegerät existieren. Relativ oder indiziert organisierte Dateien sind nur auf Plattenspeichern möglich.

Sequentielle Dateiorganisation

In einer Datei mit sequentieller Dateiorganisation hat jeder Satz, außer dem letzten, einen eindeutigen Nachfolgesatz, und jeder Satz, außer dem ersten, einen eindeutigen Vorgängersatz. Diese Vorgänger- bzw. Nachfolgereigenschaft wird beim Aufbau der Datei durch die Reihenfolge der WRITE-Anweisungen festgelegt. Diese Eigenschaft bleibt während der Lebensdauer einer Datei erhalten. Die Datei kann lediglich an ihrem Ende fortgesetzt werden.

Eine sequentiell organisierte Plattenspeicherdatei hat die gleiche logische Struktur wie jede andere sequentielle Datei auf anderen Speichermedien. Jedoch kann eine Plattenspeicherdatei ohne Kopiervorgänge am ursprünglichen Platz aktualisiert werden. Beim Aktualisieren einer Plattenspeicherdatei mit sequentieller Organisation können jedoch keine neuen Sätze eingefügt werden, sondern nur bereits existierende ersetzt werden, wobei der neue Satz

gleiche Satzlänge wie der bereits existierende haben muß. Im Eröffnungsmodus ›EXTEND‹ kann die Datei verlängert werden.

Relative Dateiorganisation

Bei einer Datei mit relativer Dateiorganisation kann zu jedem Satz dieser Datei zugegriffen werden, durch Angabe der relativen Satznummer dieses Satzes in der Datei. Vom Konzept her besteht eine Datei mit relativer Dateiorganisation aus einer Folge von Bereichen, in denen jeweils ein logischer Satz der Datei abgespeichert werden kann. Jedem dieser Bereiche ist eine relative Satznummer zugeordnet, die damit automatisch auch den logischen Satz identifiziert. Ein Satz mit der relativen Satznummer 1Ø ist der 1Øte Satz der Datei und befindet sich im 1Øten Satzspeicherbereich, unabhängig davon, ob die Sätze 1—9 bereits existieren oder nicht.

Indizierte Dateiorganisation

Bei einer Datei mit indizierter Organisation geschieht der Zugriff zu jedem Satz der Datei über den Wert eines im Satz enthaltenen Schlüssels. Dabei wird für jeden Schlüsselwert, der in der Datei vorkommt, ein Index mitgeführt. Diese Indizes stellen damit einen Zugriffsmechanismus zu jedem Satz der Datei dar. Die in der RECORD KEY-Klausel angegebenen Schlüsselwerte, die zum Aufbau der Datei verwendet werden, müssen eindeutig sein.

Satzformate:

Für jede Organisationsform sind bestimmte Satzformate erlaubt; dies zeigt die folgende Tabelle.

Tabelle: Satzformate in Abhängigkeit von der Organisationsform

Organisationsform	Satzformate	
	geblockt	ungeblockt
sequentiell	F, V	F, V, U
relativ	F	
indiziert	F, V	F, V

Zugriffsarten

Außer der Organisationsform bestimmt auch die Zugriffsart die Verarbeitung einer Datei. Die Zugriffsart legt der Benutzer mit der ACCESS-Klausel fest. Möglich sind:

- sequentieller Zugriff (ACCESS IS SEQUENTIAL)
- wahlfreier Zugriff (ACCESS IS RANDOM)
- dynamischer Zugriff (ACCESS IS DYNAMIC)

Die folgende Tabelle gibt einen Überblick über die Zugriffsarten und ihre Funktionen, die für die einzelnen Organisationsformen erlaubt sind.

Tabelle: Zugriffsart in Abhängigkeit von der Organisationsform

Organisationsform ORGANIZATION- Klausel	Zugriffsmodus ACCESS- Klausel	Bemerkungen
SEQUENTIAL (Standard- annahme)	SEQUENTIAL oder keine Angabe	Zugriff erfolgt entsprechend der Reihenfolge in der die Sätze in die Datei geschrieben werden.
RELATIVE	SEQUENTIAL oder keine Angabe	Zugriff erfolgt in aufsteigender Reihenfolge der relativen Satznummer. Nur bereits in der Datei existierende Sätze werden zur Verfügung gestellt. Mit Hilfe der START-Anweisung kann der Beginn des sequentiellen Zugriffs angegeben werden.
	RANDOM	Zugriff erfolgt in der vom Benutzer durch die im RELATIVE KEY zur Verfügung gestellten Satznummer in der vom Benutzer gewünschten Reihenfolge.
	DYNAMIC	Bei dieser Angabe kann der Anwender durch Verwendung geeigneter COBOL-Anweisungen zwischen sequentiellem und wahlfreiem Zugriff während der Verarbeitung einer Datei wählen.
INDEXED	SEQUENTIAL oder keine Angabe	Zugriff erfolgt in aufsteigender Reihenfolge der Satzschlüssel. Nur bereits in der Datei existierende Sätze werden zur Verfügung gestellt. Mit Hilfe der START-Anweisung kann der Beginn des sequentiellen Zugriffs festgelegt werden.
	RANDOM	Zugriff erfolgt in der vom Benutzer durch den im RECORD KEY zur Verfügung gestellten Satzschlüssel in der von ihm gewünschten Reihenfolge.
	DYNAMIC	Bei dieser Angabe kann der Anwender durch Verwendung geeigneter COBOL-Anweisungen zwischen sequentiellem und wahlfreiem Zugriff während der Verarbeitung einer Datei wählen.

Dateieröffnung

Die Anweisung

```
     ⎧ INPUT . . . [REVERSED] ⎫
OPEN ⎨ OUTPUT                 ⎬ . . .
     ⎪ EXTEND                 ⎪
     ⎩ I-O                    ⎭
```

eröffnet eine Datei. Abhängig von der Eröffnungsart und der Organisationsform sind bestimmte Verarbeitungsarten für die Eingabe bzw. Ausgabe zulässig. Die folgende Tabelle zeigt dies.

Eröffnungsart / Organisationsform	INPUT	OUTPUT	EXTEND	I-O	INPUT . . . REVERSED
SEQUENTIAL	READ	WRITE	WRITE	READ, REWRITE[1])	READ
RELATIVE	READ START	WRITE DELETE	–	READ, START WRITE, REWRITE, DELETE	–
INDEXED	READ START	WRITE DELETE	–	READ, START WRITE, REWRITE, DELETE	–

[1]) Nur für Plattenspeicher-Dateien erlaubt

Tabelle: Verarbeitungsarten in Abhängigkeit von Eröffnungsart und Organisationsform

Die im OPEN angegebene Eröffnungsart wird nicht berücksichtigt, wenn vor Start des Programms ein FILE-Kommando für dieselbe Datei mit OPEN-Operand gegeben wurde. Das bedeutet, daß für eine Datei, die in einem Programmlauf mehrfach im Wechsel schreibend und lesend bearbeitet wird, nur ein FILE-Kommando **ohne** OPEN-Operand gegeben werden kann, da sonst der Eröffnungsmodus für den ganzen Programmlauf festgelegt wäre.

Folgende Angaben in der OPEN-Anweisung des COBOL-Programmes und im OPEN-Operanden des FILE-Kommandos sind unvereinbar:

COBOL-Anweisung	FILE-Kommando
OPEN INPUT . . . [REVERSED]	OPEN = OUTPUT OPEN = EXTEND
OPEN OUTPUT	OPEN = INPUT OPEN = REVERSE
OPEN EXTEND	OPEN = INPUT OPEN = REVERSE

APPLY BLOCK-DENSITY-Klausel

Die Klausel ist nur bei ISAM-Dateien von Bedeutung. Sie bestimmt den Füllungsgrad der Datenblöcke. Bei Angabe von z. B. 60% werden nur soviele Sätze in einem ISAM-Datenblock eingefügt, wie zur Belegung von 60% des für den Block zur Verfügung stehenden Speicherplatzes erforderlich sind. Die restlichen 40% bleiben für spätere Aktualisierungsvorgänge zur Verfügung. Diese Angabe ist jedoch nur dann sinnvoll, falls beim Laden der Datei Lücken in der aufsteigenden Folge der Schlüssel der Datensätze vorhanden sind.

Beispiel:
Es soll eine ISAM-Datei geladen werden, wobei nur jeder 10-te Satz zur Verfügung steht. Um beim späteren Aktualisieren der Datei die Erzeugung von Überlauf-Blöcken soweit wie möglich auszuschließen, ist es sinnvoll, die Datei mit der Klausel APPLY BLOCK-DENSITY 10 PERCENT zu laden.
Beim späteren Einfügen der jeweils fehlenden 9 Sätze pro Block erfolgt dann nur eine Verschiebung innerhalb der bereits vorgegebenen Blöcke, ohne daß Überlauf-Blöcke gebildet werden müssen. Die Anwendung dieser Klausel erhöht also die Effizienz des Abarbeitens einer ISAM-Datei, vor allem nach vielen Aktualisierungsvorgängen; bedeutet allerdings einen erhöhten Platzbedarf der Datei, falls mit vielen Lücken erstmalig geladen wird.

Übertragung der Daten

Um auf Datensätze in einer SAM- oder ISAM-Datei zuzugreifen, verwendet das COBOL-Programm eines der beiden folgenden Verfahren (siehe [4]):

- **Move Mode** (Übertragungsbetrieb):
  Bei jedem Zugriff überträgt das DVS einen logischen Satz zwischen dem Puffer und einem Arbeitsbereich des Programmes.
- **Locate Mode** (Ortungsbetrieb):
  Bei jedem Zugriff übergibt das DVS dem Programm eine Adresse, die auf eine Stelle im Puffer zeigt, von der das Programm den angeforderten Satz lesen oder wohin es einen Satz schreiben kann. Ein Register zur Übergabe dieser Adresse wird beim Eröffnen der Datei vereinbart.

Für den Zugriff auf Daten in relativen Dateien benutzt das COBOL-Programm einen speziellen COBOL Move Mode.

Welche dieser Übertragungsarten das Programm jeweils auswählt, hängt ab von der Organisationsform der Datei, dem Format ihrer Sätze und der Eröffnungsart. Die folgende Tabelle gibt darüber Aufschluß.

Organisationsform \ Eröffnungsart		OUTPUT	INPUT	I-O
SEQUENTIAL	RECFORM = V RECFORM = F RECFORM = U	Move Mode Locate Mode Locate Mode	Locate Mode Locate Mode Locate Mode	Locate Mode Locate Mode Locate Mode
INDEXED	RECFORM = V RECFORM = F	Move Mode Move Mode	Move Mode Move Mode	Move Mode Move Mode
RELATIVE		COBOL Move Mode	COBOL Move Mode	COBOL Move Mode

Tabelle: Übertragungsarten für Daten in Abhängigkeit von der Eröffnungsart und der Organisationsform der Datei

Hinweis: Im Locate Mode liegt der Satzbereich einer Datei im Puffer. Bei geblockten SAM-Dateien, die im Modus I-O eröffnet sind, wird daher durch eine REWRITE-Anweisung für einen Satz eines Blockes stets der gesamte Block übertragen. Darauf ist insbesondere dann zu achten, wenn Sätze eines Blockes zwar modifiziert wurden, die Änderungen jedoch nicht zurückgeschrieben werden sollen: Sie werden implizit durch eine REWRITE-Anweisung für einen anderen Satz dieses Blockes mit übertragen.

6.5.3 Sequentielle Dateiorganisation

Eine sequentiell organisierte Datei kann nur so verarbeitet werden, daß Sätze in der Reihenfolge gelesen oder geschrieben werden, wie sie in der Datei vorkommen. Das BS2000 stellt dazu die Zugriffsmethode SAM zur Verfügung.

Die folgende Abbildung gibt die COBOL-Klauseln wieder, die man zum sequentiellen Zugriff verwenden kann.

Tabelle: COBOL-Klauseln für sequentielle Dateiorganisation

DVS Zugriffs-methode	Geräte-typ	ACCESS-Klausel	KEY-Klausel	OPEN-Anweisung	Ein-Ausgabe-Anweisung	CLOSE-Anweisung
SAM	Band	SEQUENTIAL	nicht zulässig	INPUT [REVERSED NO REWIND]	READ [INTO] [AT END]	[REEL] [LOCK NO REWIND FOR REMOVAL]
				OUTPUT [NO REWIND]	WRITE [FROM]	
				EXTEND	WRITE [FROM]	
	Platten-speicher	SEQUENTIAL		INPUT	READ [INTO] [AT END]	[UNIT] [LOCK NO REWIND]
				OUTPUT	WRITE [FROM]	
				EXTEND	WRITE [FROM]	
				I-O	READ [INTO] [AT END] WRITE [FROM] REWRITE [FROM]	[LOCK]

Steuerung des Programmablaufs

Das FILE-Kommando definiert Dateieigenschaften zur Ablaufzeit eines COBOL-Programms. Für sequentielle Dateien sind besonders vier Operanden wichtig; ausführliche Beschreibung siehe „Kommandosprache", [2]:

Format des FILE-Kommandos:

```
                                               { STD         }           { p     }
/FILE dateiname[,LINK = linkname][,BLKSIZE =   { (STD,n)     } ][,SPACE = { (p,s) } ][,CODE = ISO7]
                                               { pufferlänge }
```

dateiname	Name, unter dem die Datei katalogisiert ist
LINK=linkname	Dateikettungsname
BLKSIZE=STD	Der Puffer für die Ein-Ausgabe der Datei wird auf die Größe eines Standardblockes (1 PAM-Seite = 2048 Bytes) festgelegt.
(STD,n)	Der Puffer für die Ein-Ausgabe der Datei wird auf die Größe von n Standardblöcken festgelegt. n ist eine Zahl im Bereich von 2—16
pufferlänge	Die Pufferlänge wird in Byte angegeben. Die Angabe darf maximal 32767 sein. (Diese Angabe ist nur für MB-Dateien möglich.)
Hinweis:	Der Operand BLKSIZE=(STD,n) ermöglicht für n>1 gekettete Ein-Ausgabe (chained I-O): Intern werden mit nur einem Makroaufruf bis zu 16 logisch aufeinanderfolgende PAM-Blöcke übertragen. Die Anzahl der Ein-Ausgabe-Operationen wird damit verringert, die Verarbeitung beschleunigt (ab BS2000 V6.0).

SPACE=p	p ist die Anzahl der PAM-Seiten für die Primärzuweisung.
(p,s)	s ist die Anzahl der PAM-Seiten, für die Sekundärzuweisung. Beide Angaben sollten ein Vielfaches von 3 sein, da die PAM-Seitenzuweisung in Einheiten (units) von 3 erfolgt. Ist die Angabe kein Vielfaches von 3, wird sie auf einen entsprechenden Wert aufgerundet.
Hinweis:	Bei BLKSIZE = (std,n) gilt für — die Primärzuweisung p ≥ 2 × n — und die Sekundärzuweisung s ≥ n (Primär- und Sekundärzuweisung jeweils aufgerundet auf das Vielfache von 3) Beispiel: BLKSIZE=(STD,8) SPACE=(18,9)
CODE=ISO7	Magnetbanddateien im ISO-7-Bit-Code (s. u.) werden verarbeitet. Bei fehlender Angabe verwendet das System den EBCDIC-Code (Standard).

Hinweis: Bei Daten mit RECFORM = U (RECORDING U-Angabe in der FILE SECTION) darf im FILE-Kommando der RECSIZE-Operand nicht angegeben werden. In diesem Fall bezeichnet RECSIZE ein Register zur Übergabe der Satzlängeninformation.

Steuerinformationen für Druckerdateien

Die vertikale Positionierung einer Druckerdatei, bei der der Gerätename **PRINTERDOD** angegeben wurde, ist auf zwei Arten möglich:

1. WRITE ohne ADVANCING-Zusatz
2. WRITE mit ADVANCING-Zusatz

In beiden Fällen dient das erste Zeichen des benutzerdefinierten Satzes zur Übergabe der Vorschubinformation an das DVS und muß vom Anwender reserviert werden.

Bei Verwendung von WRITE ohne ADVANCING-Zusatz muß der Anwender das erste Byte des Satzes selber mit der gewünschten Vorschubinformation versorgen. Die möglichen Werte sind in den folgenden Tabellen abgedruckt.

Tabelle: Steuerinformation für Druckervorschub um Zeilenanzahl

Vorschub um Anzahl Zeilen	Inhalt des Steuerbyte falls Vorschub nach	vor	Drucken erwünscht
1	Ø1	4Ø	Die Werte des zweiten Halbbyte für das Steuerzeichen Vorschub vor dem Drucken sind wegen Hardwareeigenschaften um 1 kleiner, als die gewünschte Zeilenzahl.
2	Ø2	41	
3	Ø3	42	
4	Ø4	43	
5	Ø5	44	
6	Ø6	45	
.	.	.	
.	.	.	
.	.	.	
9	Ø9	48	
1Ø	ØA	49	
11	ØB	4A	
12	ØC	4B	
13	ØD	4C	
14	ØE	4D	
15	ØF	4E	

Tabelle: Steuerinformation für Druckervorschub nach Lochbandkanälen.

Vorschub nach Lochband-Kanälen	Vorschub nach dem Drucken	Vorschub vor dem Drucken
1 (Seitenwechsel)	81	C1
2	82	C2
3	83	C3
4	84	C4
5	85	C5
6	86	C6
7	87	C7
8	88	C8
10*)	8A	CA
11	8B	CB

*) Ein Vorschub nach Kanal 9 oder 12 ist nicht möglich, da diese nur zu Formularendebestimmung dienen.

Bei den aufgeführten Steuerzeichen handelt es sich nur in einigen Fällen um abdruckbare Zeichen. In folgenderTabelle sind die abdruckbaren Zeichen und ihre Äquivalente aufgeführt.

Steuerzeichen	abdruckbares Zeichen
4Ø	␣ (Zwischenraum)
4A	c (cent)
4B	. (Punkt)
4C	< (kleiner als)
4D	((Klammer auf)
4E	+ (Plus)
C1	A
C2	B
C3	C
C4	D
C5	E
C6	F
C7	G
C8	H

Druckerdateien werden mit dem PRINT-Kommando ausgedruckt. Es muß der Operand SPACE=E angegeben werden. Genaue Beschreibung und Erläuterung des PRINT-Kommandos siehe [2, Kapitel 3].

Im folgenden wird eine Möglichkeit zur Erzeugung eines beliebigen sedezimalen Zahlenwertes angegeben.

Beispiel 39:

Es soll der sedezimale Wert 0A erzeugt und in das Steuerzeichen des Drucksatzes übertragen werden:

```
    .
    .
    .
    .
SPECIAL-NAMES.
    .
    .
    .
    .
    SYMBOLIC CHARACTERS HEX-0A IS 11
    .
    .
    .
    .
01  SATZ.
    02 STEUERZEICHEN  PIC X.
    02 DATEN          PIC X (132).
    .
    .
    .
    .
    MOVE HEX-0A TO STEUERZEICHEN.
    .
    .
    .
    .
```

Erläuterung:

Dem Datennamen HEX-0A wird das elfte Zeichen des EBCDIC-Zeichensatzes zugeordnet, nämlich der sedezimale Wert 0A.

Mit der MOVE-Anweisung wird dieser Wert in das Feld STEUERZEICHEN übertragen.

ISO-7-Bit CODE

Um Magnetbanddateien im ISO-7-Bit Code zu verarbeiten, gibt es zwei Möglichkeiten:

1. Angabe des Gerätenamens TAPISO im Herstellerwort der SELECT-Klausel.

   Außerdem ist vor Programmablauf der Operand CODE=ISO7 im FILE-Kommando erforderlich:

   ```
   /FILE dateiname,CODE=ISO7, . . .
   ```

2. Im SPECIAL-NAMES-Paragraphen verwendet man die Klausel

   ```
   ALPHABET alphabet-1 IS STANDARD-2
   ```

   und in der FD-erklärung der Magnetbanddatei

   ```
   CODE-SET IS alphabet-1
   ```

Ein-Ausgabe-Zustände

Gibt der Benutzer in der SELECT-Klausel "FILE STATUS IS datenname" an, so wird der Ein-Ausgabe-Zustand der zugehörigen Datei nach jeder Ein-Ausgabeoperation im Feld "datenname" abgelegt. Dieses Feld fragt der Benutzer ab, um Ein-Ausgabeoperationen zu kontrollieren. Die für eine sequentielle Datei möglichen Werte sind nachfolgend zusammengestellt.

Tabelle: Ein-Ausgabe-Zustände

FILE STATUS-Werte	Bedeutung
0×	erfolgreiche Ausführung
00	keine weitere Information
04	erfolgreicher READ, aber Satzlängenfehler (siehe COB1 Beschreibung [1]: RECORD-Klausel)
05	erfolgreicher OPEN INPUT oder OPEN I-O auf OPTIONAL-Datei, die nicht vorhanden ist; einer der folgenden Gründe liegt vor: 1. Das FILE-Kommando mit richtigem Linknamen fehlt; der Programmlauf wird mit der Meldung 9077 unterbrochen; bei OPEN INPUT/I-O wird das FILE-Kommando angefordert; 2. Datei ist nicht katalogisiert; 3. Datei ist zwar katalogisiert, aber OPEN und CLOSE wurden noch nicht ausgeführt.
1×	erfolglose Ausführung: Endebedingung
10	READ bei Dateiende
15	erster READ auf eine nicht vorhandene OPTIONAL-Datei; Dateiende erreicht (AT END-Bedingung)
16	READ nach bereits erkannter AT END-Bedingung
3×	erfolglose Ausführung: permanenter Fehler
30	keine weitere Information
34	unzureichende Sekundärzuweisung im FILE-Kommando (aufgrund der Zwischenpufferung ist die Anzahl der tatsächlich geschriebenen Sätze unbestimmt)
35	OPEN INPUT/I-O auf eine nicht vorhandene Datei ohne OPTIONAL-Angabe; einer der folgenden Gründe liegt vor: 1. Das FILE-Kommando mit richtigem Linknamen fehlt; der Programmlauf wird mit der Meldung 9077 unterbrochen; bei OPEN INPUT/I-O wird das FILE-Kommando angefordert; 2. Datei ist nicht katalogisiert; 3. Datei ist zwar katalogisiert, aber OPEN und CLOSE wurden noch nicht ausgeführt.
38	OPEN auf eine Datei, die vorher mit CLOSE WITH LOCK geschlossen wurde
39	1. Im FILE-Kommando wurden einer oder mehrere der Operanden FCBTYPE, RECFORM oder RECSIZE mit Werten angegeben, die von den entsprechenden expliziten oder impliziten Programmangaben abweichen. 2. Satzlängenfehler bei Eingabedateien (Katalogüberprüfung). 3. Satzlänge größer als BLKSIZE-Angabe im FILE-Kommando.
4×	erfolglose Ausführung: logischer Fehler
41	OPEN auf eine bereits eröffnete Datei
42	CLOSE auf eine nicht eröffnete Datei
43	REWRITE ohne vorherigen erfolgreichen READ
44	Überschreiten der Bereichsgrenzen: 1. WRITE oder REWRITE mit unzulässiger Satzlänge (siehe COB1 Beschreibung [1]: RECORD-Klausel) 2. REWRITE mit anderer Satzlänge als beim gelesenen Satz
46	erneuter READ nach erfolglosem READ
47	READ auf eine Datei, die nicht als INPUT oder I-O eröffnet wurde
48	WRITE auf eine Datei, die nicht als OUTPUT, I-O oder EXTEND eröffnet wurde
49	REWRITE auf eine Datei, die nicht als I-O eröffnet wurde
9×	sonstige erfolglose Ausführungen
90	Systemfehler; keine weitere Information
91	entweder Password-Fehler oder OPEN-Fehler oder kein freies Gerät

Leerseite durch den Nachtrag vom August 1986

6.5.4 Relative Dateiorganisation

Die Verarbeitung relativ organisierter Dateien wird bestimmt durch die Verwendung der relativen Satzadressierung. Unter Verwendung dieser Adressierungsmöglichkeit wird die Position jedes logischen Satzes innerhalb der Datei relativ zum ersten Satz dieser Datei festgelegt, und zwar durch eine relative Satznummer, beginnend mit 1. Durch Angabe eines relativen Schlüsselfeldes kann der Benutzer direkt auf Sätze einer relativen Datei zugreifen.

Relativ organisierte Dateien sind nur auf Plattenspeichern möglich und können nur aus Sätzen fester Länge bestehen.

Im BS2000 wird die Zugriffsmethode PAM benutzt.

Die folgende Tabelle gibt die COBOL-Klauseln wieder, die für den Zugriff zu relativ organisierten Dateien verwendet werden können.

DVS-Zugriffsmethode	Gerätetyp	ACCESS-Klausel	KEY-Klausel	OPEN-Anweisung	Ein-Ausgabe-Anweisungen	CLOSE-Anweisung
PAM	Plattenspeicher	SEQUENTIAL	RELATIVE KEY	INPUT OUTPUT I—O	READ [WITH NO LOCK] [INTO] [AT END] START [WITH NO LOCK] WRITE [FROM] READ [WITH NO LOCK] [INTO] [AT END] START [WITH NO LOCK] WRITE [FROM] REWRITE [FROM] DELETE	[WITH LOCK]
		RANDOM	RELATIVE KEY	INPUT OUTPUT I—O	READ [WITH NO LOCK] [INTO] [INVALID KEY] WRITE [FROM] INVALID KEY READ [WITH NO LOCK] [INTO] [INVALID KEY] WRITE [FROM] [INVALID KEY] REWRITE [FROM] [INVALID KEY] DELETE [INVALID KEY]	[WITH LOCK]
		DYNAMIC	RELATIVE KEY	INPUT OUTPUT I—O	READ [WITH NO LOCK] NEXT [INTO] [AT END] READ [WITH NO LOCK] [INTO] [INVALID KEY] START [WITH NO LOCK] [INVALID KEY] WRITE [FROM] [INVALID KEY] READ [WITH NO LOCK] NEXT [INTO] [AT END] READ [WITH NO LOCK] [INTO] [INVALID KEY] START [WITH NO LOCK] [INVALID KEY] WRITE [FROM] [INVALID KEY] REWRITE [FROM] [INVALID KEY] DELETE [INVALID KEY]	[WITH LOCK]

Aufbau einer relativen Datei

Jede PAM-Seite einer relativen Datei beginnt mit einem PAM-Schlüssel (Aufbau siehe Manual „DVS, Plattenverarbeitung" [4]), gefolgt von logischen Sätzen. Für Ausgabedateien kann mit dem BLKSIZE-Operanden im FILE-Kommando die Blockgröße geändert werden, die im COBOL-Programm durch die BLOCK-CONTAINS-Klausel festgelegt wurde. Das RTS (Runtime-System) macht aus der Blockgröße des COBOL-Programms eine Standard-Blockgröße.

Beispiel: COBOL-Blockgröße 1000 Characters → (STD,1)
5000 Characters → (STD,3)

Arbeitet man mit SHARUPD=YES, so wird bei Zugriffen mit Sperrmechanismus der gesamte Block gegen den Zugriff simultaner Benutzer gesperrt.

Die folgende Abbildung zeigt den Aufbau einer relativen Datei für einen Block mit 2 PAM-Seiten; BLKSIZE=(STD,2).

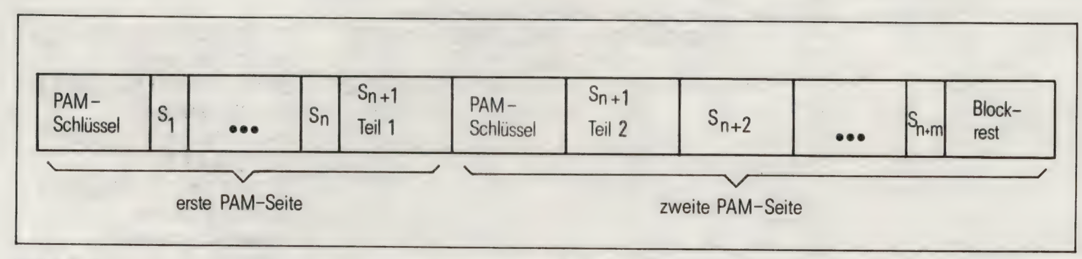


Bild 6-3

Dem PAM-Schlüssel auf der ersten PAM-Seite folgen die Sätze S_1 bis S_n. Der vollständige Satz S_{n+1} hat auf der ersten PAM-Seite nicht mehr Platz. Ein Teil wird deshalb auf die zweite PAM-Seite geschrieben. Am Ende des Blockes bleibt möglicherweise Platz übrig (Blockrest).

Eine relative Datei muß vor ihrer Erstellung nicht vorformatiert werden. Erstellen kann man im sequentiellen oder wahlfreien Zugriff.

Bei sequentieller Erstellung wird der RELATIVE KEY nicht ausgewertet. Die Sätze werden in der Reihenfolge in die Datei gebracht, wie sie vom Benutzer geschrieben werden (WRITE). Leersätze kann es hierbei nur im letzten Block geben.

Bei wahlfreier Erstellung errechnet das Ablaufzeitsystem aufgrund der definierten RECSIZE, BLKSIZE (FILE-Kommando) und des Inhalts des RELATIVE KEY die genaue Position des Satzes innerhalb der Datei. Dabei wird jeder angefangene Block vorformatiert.

Beispiel 43: Programm zur Einrichtung einer relativen Datei

```
ID DIVISION.
PROGRAM-ID. RELDAT.
ENVIRONMENT DIVISION.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT AUS ASSIGN TO DA-590-R-SYS010                    ①
    ACCESS IS DYNAMIC
    RELATIVE KEY IS REL-SCHLUESSEL.                          ②
DATA DIVISION.
FILE SECTION.
FD AUS LABEL RECORD STANDARD.                                ③
01 AUSSATZ PIC X(200).
WORKING-STORAGE SECTION.
01 EINGABE.
  05 ANF       PIC X(5) VALUE SPACES.
  05 REL-SCHLUESSEL   PIC 9(10) VALUE ZEROES.                ③
  05 RE        PIC X(185).
01 FE.
 02 F OCCURS 6 TIMES INDEXED BY I.
  05 FELD PIC X(185).
PROCEDURE DIVISION.
    OPEN OUTPUT AUS.
    MOVE ALL "1" TO F(01).
    MOVE ALL "2" TO F(02).
    MOVE ALL "3" TO F(03).
    MOVE ALL "4" TO F(04).
    MOVE ALL "5" TO F(05).
    MOVE ALL "6" TO F(06).
    SET I TO 1.
ANFANG.
    ADD 1 TO REL-SCHLUESSEL.
    IF REL-SCHLUESSEL = 3 ADD 1 TO REL-SCHLUESSEL.
    IF REL-SCHLUESSEL = 6 GO TO ENDE.
    MOVE F(I) TO RE.
    MOVE EINGABE TO AUSSATZ.
    SET I UP BY 1.
    WRITE AUSSATZ INVALID KEY
      DISPLAY "SCHLUESSEL-FEHLER   " UPON TERMINAL
      GO TO ENDE.
    GO TO ANFANG.
ENDE.
    CLOSE AUS.
    DISPLAY "RELDAT BEENDET" UPON TERMINAL.
    STOP RUN.
```

① R bezeichnet „relative Datei".

② Die RELATIVE KEY-Klausel wird verwendet.

③ Im Gegensatz zu anderen COBOL-Schlüsseln wird der relative Satzschlüssel nicht in der FD beschrieben, sondern in der WORKING-STORAGE SECTION.

Mit Hilfe des Dienstprogramms DPAGE [3] läßt sich der Inhalt einer relativen Datei ausdrucken. Im Beispiel 44 handelt es sich um die mit obigem Programmbeispiel beschriebene Datei.

Bei sequentiellem Zugriff zu einer relativ organisierten Datei wird die relative Satznummer des zuletzt gelesenen oder geschriebenen Satzes dem Benutzer im relativen Satzschlüssel zur Verfügung gestellt.

Bei direkter Verarbeitung muß der Benutzer dem System im relativen Satzschlüssel die Nummer des Satzes übergeben, der gelesen, geschrieben, gelöscht oder verändert werden soll, bzw. auf den der Benutzer positionieren will.

Steuerung des Programmablaufs

Das FILE-Kommando definiert Dateieigenschaften zur Ablaufzeit eines COBOL-Programms. Für relative Dateien sind besonders vier Operanden wichtig; ausführliche Beschreibung siehe „Kommandosprache", [2]:

Format des FILE-Kommandos:

```
/FILE dateiname [,LINK=linkname]
                [,SPACE={ p     } ]
                        { (p,s) }
                [,BLKSIZE={ STD     } ]
                          { (STD,n) }
                [,SHARUPD={ YES } ]
                          { NO  }
```

(STD und NO sind unterstrichen.)

dateiname	Der Name ist anzugeben, unter dem die Datei katalogisiert ist.
LINK =linkname	Der Dateikettungsname ist anzugeben (=Linkname=FD-Name).
SPACE={p / (p,s)}	Für p ist die Anzahl der PAM-Seiten für die Primärzuweisung als Vielfaches von 3 anzugeben. Für s ist die Anzahl der PAM-Seiten als Vielfaches von 3 anzugeben. Bei fehlenden Operanden setzt das System für p bzw. s den Standardwert 3.
BLKSIZE =STD	vereinbart einen Standardblock als Puffer, also 2048 Bytes.
=(STD,n)	vereinbart einen Pufferbereich von n Standardblöcken (PAM-Seiten). n darf maximal 16 sein (das sind 16 × 2048 Bytes = 32768 Bytes).
SHARUPD =YES	Mehrere Prozesse können die Datei gleichzeitig ändern, d. h. sie ist nicht gesperrt, sobald sie ein Prozeß im Ausgabemodus eröffnet hat; siehe Abschnitt 6.5.7.
=NO	Mehrere Prozesse können die Datei nur dann gleichzeitig verwenden, wenn sie alle im Lesemodus arbeiten.

Hinweis: Der Operand BLKSIZE=(STD,n) ermöglicht für n>1 gekettete Ein-Ausgabe (chained I-O): Intern werden mit nur einem Makroaufruf bis zu 16 logisch aufeinanderfolgende PAM-Blöcke übertragen. Die Anzahl der Ein-Ausgabe-Operationen wird damit verringert, die Verarbeitung beschleunigt.

Ein-Ausgabe-Zustände

Gibt der Benutzer in der SELECT-Klausel "FILE STATUS IS datenname" an, so wird der Ein-Ausgabe-Zustand der zugehörigen Datei nach jeder Ein-Ausgabeoperation im Feld "datenname" abgelegt. Dieses Feld fragt der Benutzer ab, um Ein-Ausgabeoperationen zu kontrollieren. Die für eine relative Datei möglichen Werte sind nachfolgend zusammengestellt.

Tabelle: Ein-Ausgabe-Zustände

FILE STATUS-Werte	Bedeutung
0×	erfolgreiche Ausführung
00	keine weitere Information
1×	erfolglose Ausführung: Endebedingung
10	READ bei Dateiende
14	Bei einer READ-Anweisung reicht das RELATIVE KEY-Feld nicht aus, um die relative Satznummer aufzunehmen.
16	READ nach bereits erkannter AT END-Bedingung
2×	erfolglose Ausführung: Schlüsselfehler
22	WRITE für bereits vorhandenen Satz
23	versuchter Zugriff auf einen nicht vorhandenen Datensatz (INVALID KEY-Bedingung)
24	Zwei Möglichkeiten: 1. Bei sequentiellem WRITE überschreitet die Satznummer die Größe des RELATIVE KEY-Feldes. 2. unzureichende Sekundärzuweisung im FILE-Kommando
3×	erfolglose Ausführung: permanenter Fehler
30	keine weitere Information
35	OPEN INPUT/I-O auf eine nicht vorhandene Datei; einer der folgenden Gründe liegt vor: 1. Das FILE-Kommando mit richtigem Linknamen fehlt; der Programmlauf wird mit der Meldung 9077 unterbrochen; bei OPEN INPUT/I-O wird das FILE-Kommando angefordert; 2. Datei ist nicht katalogisiert; 3. Datei ist zwar katalogisiert, aber OPEN und CLOSE wurden noch nicht ausgeführt.
38	OPEN auf eine Datei, die vorher mit CLOSE WITH LOCK geschlossen wurde
39	1. Im FILE-Kommando wurden einer oder mehrere der Operanden FCBTYPE, RECFORM oder RECSIZE mit Werten angegeben, die von den entsprechenden expliziten oder impliziten Programmangaben abweichen. (FCBTYPE muß PAM sein, RECFORM oder RECSIZE sollten überhaupt nicht angegeben werden). 2. Satzlängenfehler bei Eingabedateien (Katalogüberprüfung). 3. Satzlänge größer als BLKSIZE-Angabe im FILE-Kommando.
4×	erfolglose Ausführung: logischer Fehler
41	OPEN auf eine bereits eröffnete Datei
42	CLOSE auf eine nicht eröffnete Datei
43	DELETE oder REWRITE ohne vorherigen erfolgreichen READ bei ACCESS SEQUENTIAL
46	sequentieller READ nach erfolglosem READ oder START
47	READ/START auf eine Datei, die nicht als INPUT oder I-O eröffnet wurde
48	Es wurde versucht, eine WRITE-Anweisung auszuführen: 1. für eine Datei, die nicht als OUTPUT oder I-O eröffnet ist 2. für eine Datei im sequentiellen Zugriffsmodus, die nicht als OUTPUT eröffnet ist.
49	DELETE oder REWRITE auf eine Datei, die nicht mit OPEN I-O eröffnet wurde
9×	sonstige erfolglose Ausführungen
90	Systemfehler; keine weitere Information
91	entweder Password-Fehler oder OPEN-Fehler
93	Fehler bei simultaner Aktualisierung (PAGE LOCK bei SHARUPD=YES)
94	Fehler bei simultaner Aktualisierung: die Aufrufsequenz READ — REWRITE/DELETE wurde nicht eingehalten

Leere Seite seit Version 2.2A

Beispiel 44: **Inhalt einer relativen Datei**

```
LOGICAL PAGE = 1

KEY -- (0001)  (000) 8558F824 01000001 00000000 00000000  ①                                            e.8.............

(0001)  (000) 40404040 40F0F0F0 F0F0F0F0 F0F0F1F1  (010) F1F1F1F1 F1F1F1F1 F1F1F1F1 F1F1F1F1       000000000111111111111111111  ②
(0033)  (020) F1F1F1F1 F1F1F1F1 F1F1F1F1 F1F1F1F1  (030) F1F1F1F1 F1F1F1F1 F1F1F1F1 F1F1F1F1  11111111111111111111111111111111
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(0193)  (0C0) F1F1F1F1 F1F1F1F1 40404040 40F0F0F0  (0D0) F0F0F0F0 F0F0F2F2 F2F2F2F2 F2F2F2F2  11111111     0000000002222222222  ③
(0225)  (0E0) F2F2F2F2 F2F2F2F2 F2F2F2F2 F2F2F2F2  (0F0) F2F2F2F2 F2F2F2F2 F2F2F2F2 F2F2F2F2  22222222222222222222222222222222
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(0385)  (180) F2F2F2F2 F2F2F2F2 F2F2F2F2 F2F2F2F2  (190) FF000000 00000000 00000000 00000000  2222222222222222 ...............  ④
(0417)  (1A0) 00000000 00000000 00000000 00000000  (1B0) 00000000 00000000 00000000 00000000  ................................
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(0577)  (240) 00000000 00000000 00000000 00000000  (250) 00000000 00000000 40404040 40F0F0F0  ........................     000
(0609)  (260) F0F0F0F0 F0F0F4F3 F3F3F3F3 F3F3F3F3  (270) F3F3F3F3 F3F3F3F3 F3F3F3F3 F3F3F3F3  00000043333333333333333333333333  ⑤
(0641)  (280) F3F3F3F3 F3F3F3F3 F3F3F3F3 F3F3F3F3  (290) F3F3F3F3 F3F3F3F3 F3F3F3F3 F3F3F3F3  33333333333333333333333333333333
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(0801)  (320) 40404040 40F0F0F0 F0F0F0F0 F0F0F5F4  (330) F4F4F4F4 F4F4F4F4 F4F4F4F4 F4F4F4F4       00000000054444444444444444
(0833)  (340) F4F4F4F4 F4F4F4F4 F4F4F4F4 F4F4F4F4  (350) F4F4F4F4 F4F4F4F4 F4F4F4F4 F4F4F4F4  44444444444444444444444444444444
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(0993)  (3E0) F4F4F4F4 F4F4F4F4 FF000000 00000000  (3F0) 00000000 00000000 00000000 00000000  44444444 .......................
(1025)  (400) 00000000 00000000 00000000 00000000  (410) 00000000 00000000 00000000 00000000  ................................
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(1185)  (4A0) 00000000 00000000 00000000 00000000  (4B0) FF000000 00000000 00000000 00000000  ................ ...............
(1217)  (4C0) 00000000 00000000 00000000 00000000  (4D0) 00000000 00000000 00000000 00000000  ................................
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(1377)  (560) 00000000 00000000 00000000 00000000  (570) 00000000 00000000 FF000000 00000000  ........................ .......
(1409)  (580) 00000000 00000000 00000000 00000000  (590) 00000000 00000000 00000000 00000000  ................................
           NEXT 5 LINES ARE IDENTICAL TO ABOVE LINE.
(1601)  (640) FF000000 00000000 00000000 00000000  (650) 00000000 00000000 00000000 00000000   ...............................
(1633)  (660) 00000000 00000000 00000000 00000000  (670) 00000000 00000000 00000000 00000000  ................................
           NEXT 4 LINES ARE IDENTICAL TO ABOVE LINE.
(1793)  (700) 00000000 00000000 FF000000 00000000  (710) 00000000 00000000 00000000 00000000  ........ .......................
(1825)  (720) 00000000 00000000 00000000 00000000  (730) 00000000 00000000 00000000 00000000  ................................
           NEXT 6 LINES ARE IDENTICAL TO ABOVE LINE.
```

① PAM-Schlüssel

②③, ⑤ logische Sätze, die beschrieben wurden

④ Leersatz: Sein erstes Byte enthält High Value (X'FF'), der Rest ist mit Low Value (X'00') gefüllt.

6.5.5 Indizierte Dateiorganisation

Mit Hilfe der indizierten Dateiverarbeitung wird es ermöglicht, bestimmte Daten aus einer indiziert-sequentiell aufgebauten Datei aufzufinden. Beim Aufbau der Datei werden Sätze in aufsteigender Reihenfolge der Satzschlüssel abgespeichert. Ferner werden Indexstufen erzeugt, die beim Verarbeiten der Datei zur Lokalisierung der Datensätze dienen. Der Schlüssel ist Teil des logischen Satzes (definiert in der FD) und muß vom Anwender in der RECORD KEY-Klausel in der ENVIRONMENT DIVISION angegeben werden. Jeder Satz in einer indizierten Datei muß daher eindeutig durch seinen Satzschlüssel ansprechbar sein.

Indizierte Dateien sind nur auf Plattenspeichern möglich. Sie können festes oder variables Satzformat haben und können geblockt werden.

Im BS2000 wird die Zugriffsmethode ISAM zur Bearbeitung von indizierten Dateien verwendet.

Die folgende Tabelle gibt die COBOL-Klauseln wieder, die für den Zugriff zu indiziert organisierten Dateien verwendet werden können.

DVS-Zugriffs-methode	Geräte-typ	ACCESS-Klausel	KEY-Klausel	OPEN-Anweisung	Ein-Ausgabe-Anweisungen	CLOSE-Anweisung
ISAM	Platten-speicher	SEQUENTIAL	RECORD KEY	INPUT	READ [WITH NO LOCK] [INTO] [AT END] START [WITH NO LOCK]	[WITH LOCK]
				OUTPUT	WRITE [FROM]	
				I—O	READ [WITH NO LOCK] [INTO] [AT END] START [WITH NO LOCK] REWRITE [FROM] DELETE	
		RANDOM.	RECORD KEY	INPUT	READ [WITH NO LOCK] [INTO] [INVALID KEY]	[WITH LOCK]
				OUTPUT	WRITE [FROM] INVALID KEY	
				I—O	READ [WITH NO LOCK] [INTO] [INVALID KEY] WRITE [FROM] [INVALID KEY] REWRITE [FROM] [INVALID KEY] DELETE [INVALID KEY]	
		DYNAMIC/ EXTENDED	RECORD KEY	INPUT	READ [WITH NO LOCK] NEXT [INTO] [AT END] READ [WITH NO LOCK] [INTO] [INVALID KEY] START [WITH NO LOCK] [INVALID KEY]	[WITH LOCK]
				OUTPUT	WRITE [FROM] [INVALID KEY]	
				I—O	READ [WITH NO LOCK] NEXT [INTO] [AT END] READ [WITH NO LOCK] [INTO] [INVALID KEY] START [WITH NO LOCK] [INVALID KEY] WRITE [FROM] [INVALID KEY] REWRITE [FROM] [INVALID KEY] DELETE [INVALID KEY]	

Dateistruktur

Eine ISAM-Datei besteht aus zwei logischen Einheiten

a) den Indexeinträgen in den Indexdatenblöcken
b) den Datensätzen, die in den Datenblöcken abgespeichert sind.

Die Indexeinträge und die Datensätze können auf verschiedenen Datenträgern liegen, sofern Privatdatenträger verwendet werden.

Blockstruktur

Indexblöcke haben eine feste Länge von einer PAM-Seite (d.h. 2048 Bytes). Datenblöcke können eine Länge zwischen 1 bis 16 PAM-Seiten haben.

Die Datenblöcke enthalten die logischen Sätze des Benutzers. Die Einträge in den Indexblöcken enthalten entweder Zeiger zu den Indexblöcken niederer Indexstufen oder, im Falle des Indexblockes der niedersten Stufe, auf die Datenblöcke.

In den Indexblöcken sind die Eintragungen in aufsteigender Reihenfolge der Schlüssel physikalisch hintereinander angeordnet; deshalb werden beim Entstehen neuer Datenblöcke stets die Indexeintragungen reorganisiert.

In den Datenblöcken sind die Sätze in aufsteigender Reihenfolge der Schlüssel verkettet; die physikalische Reihenfolge ist beliebig.

Der Indexblock der höchsten Indexstufe ist immer in der Datei enthalten, auch wenn die Datei keine Datensätze enthält.

Zusätzlich zu den Zeigern enthält dieser Indexblock eine ISAM-Etikettinformation von 36 Bytes Länge, die intern von ISAM verwendet wird. Somit beträgt die Länge des ersten Indexblockes nur 2012 Bytes.

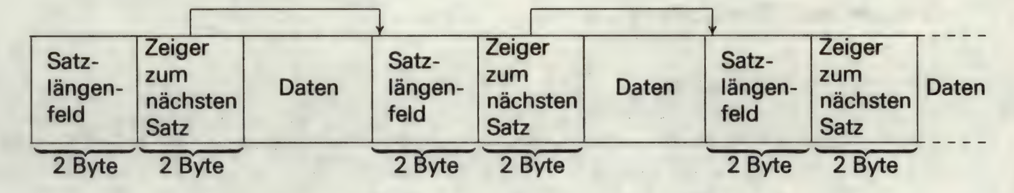
Aufbau eines Indexblockes:

Indexeintrag pro Datenblock				Indexeintrag pro Datenblock				
Nr. d. zugehörigen Datenblocks	Anzahl d. freien Bytes i. d. Block	höchster Schlüssel i. d. Block		Nr. d. zugehörigen Datenblocks	Anzahl d. freien Bytes i. d. Block	höchster Schlüssel i. d. Block		usw.
3 Byte	1 Byte	1–255 Byte		3 Byte	1 Byte	1–255 Byte		

Für jede Indexstufe wird ein Block in der Größe einer PAM-Seite im Speicher reserviert.

Verweist ein Indexeintrag auf einen Indexblock, so enthält das Byte 4 die Anzahl der Indexeinträge in diesem Block.

Aufbau eines Datenblockes:



Der Zeiger des letzten Satzes im Block zeigt auf den ersten Satz des Blockes.

Aufbau der Schlüssel

Sowohl den Indexblöcken als auch den Datenblöcken geht ein ISAM-spezifischer PAM-Schlüssel voran.

Für Indexblöcke hat er folgenden Aufbau:

Standardmäßiger Teil des PAM-Schlüssels	1. freies Byte am Ende d. Block	unbenutzt	Zeiger zum zuletzt verarbeit. Satz	unbenutzt
8 Byte	2 Byte	2 Byte	2 Byte	2 Byte

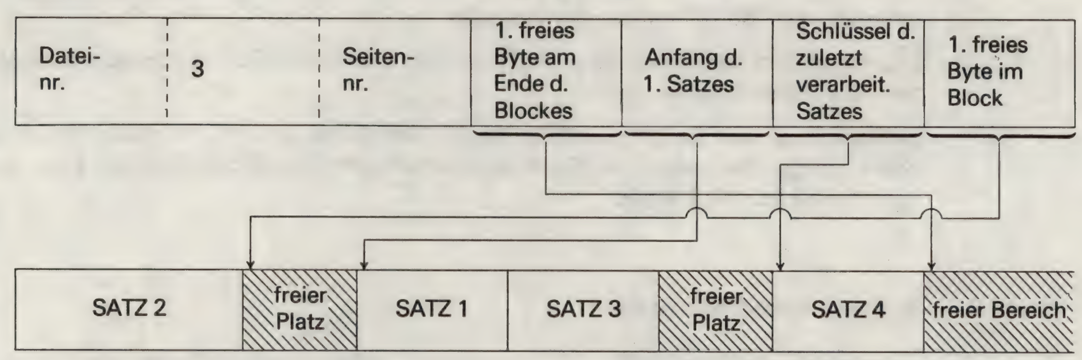
Für Datenblöcke hat er folgenden Aufbau:

Standardmäßiger Teil des PAM-Schlüssels	1. freies Byte am Ende d. Blockes	Zeiger zum 1. logischen Satz	Zeiger zum zuletzt verarbeit. Satz	1. freies Byte im Block
8 Byte	2 Byte	2 Byte	2 Byte	2 Byte

Ist die PAM-Seite nicht belegt, so ist sie in der Tabelle der freien Seiten (GARBAGE COLLECTION) eingetragen; in diesem Fall enthalten die letzten 4 Bytes des Schlüssels die Nummer der nächsten freien PAM-Seite in dieser Tabelle.

Beispiel für den PAM-Schlüssel eines Datenblockes.

PAM-Schlüssel

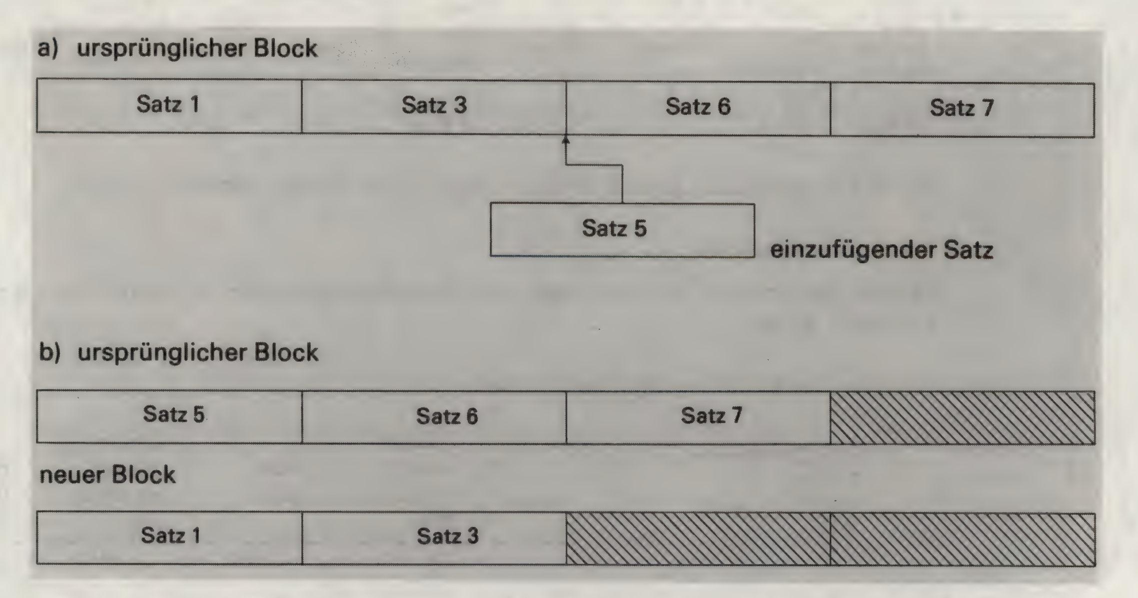


zugehörige Datensätze.

Teilen von Datenblöcken

Blockteilung tritt auf beim Ändern einer Datei, wenn z. B. Sätze in einen Block eingefügt werden sollen, im Block aber kein freier Platz mehr zur Verfügung steht. In diesem Fall wird der Block an der Stelle, an der ein Satz einzufügen ist, geteilt und die erste Hälfte des Blockinhalts in einen neuen Block geschrieben, während die zweite Hälfte einschließlich des neuen Satzes im ursprünglichen Block verbleibt und mit den Sätzen des neuen Blocks durch Zeiger verkettet wird (siehe [4]).

Beispiel 46: Blockteilung



Für den neuen Block wird gleichzeitig ein neuer Indexeintrag im zugehörigen Indexblock aufgebaut, und es werden die Indexeintragungen reorganisiert. Gleichzeitig wird die Blockreihenfolge reorganisiert, so daß niemals Überlaufkettungen entstehen.

Da eine häufige Blockteilung die Verarbeitung verlangsamt, sei hier auf die APPLY BLOCK DENSITY-Klausel hingewiesen. In dieser Klausel gibt man den Prozentsatz an, zu dem ein Datenblock beim Erstellen der Datei mit Datensätzen gefüllt werden soll. Fehlt diese Klausel, so nimmt der COB1-Übersetzer einen Füllungsgrad von 85 Prozent an. Die Blockteilung wird mit dieser Klausel eingeschränkt.

Steuerung des Programmablaufs

Das FILE-Kommando definiert Dateieigenschaften zur Ablaufzeit eines COBOL-Programms. Für indizierte Dateien sind besonders fünf Operanden wichtig; ausführliche Beschreibung siehe „Kommandosprache", [2]. Format des FILE-Kommandos:

```
/FILE dateiname [,LINK=linkname]
                [,SPACE={ p     }]
                        { (p,s) }
                [,SHARUPD={ YES }]
                          { NO  }
                [,PAD=prozent]
                [,BLKSIZE={ STD     }]
                          { (STD,n) }
```

(NO und STD sind unterstrichen.)

dateiname — Der Name ist anzugeben, unter dem die Datei katalogisiert ist.

LINK=linkname — Der Dateikettungsname ist anzugeben.

SPACE={p | (p,s)} — Für p ist die Anzahl der PAM-Seiten für die Primärzuweisung als Vielfaches von 3 anzugeben. Für s ist die Anzahl der PAM-Seiten für die Sekundärzuweisung als Vielfaches von 3 anzugeben.
Bei fehlenden Operanden setzt das System für p bzw. s den Standardwert 3.

SHARUPD =YES — Mehrere Prozesse können die Datei gleichzeitig ändern, d. h., sie ist nicht gesperrt, sobald ein Prozeß die Datei im Ausgabemodus eröffnet hat; siehe Abschnitt 6.5.7.

=<u>NO</u> — Mehrere Prozesse können die Datei nur dann gleichzeitig verwenden, wenn sie alle im Lesemodus arbeiten.

PAD=prozent — Für prozent ist eine Zahl einzusetzen; sie gibt den prozentualen Platzanteil der Pufferlänge an, der frei bleiben soll (für spätere Dateierweiterung). Standardwert: 15.

BLKSIZE

=<u>STD</u> — vereinbart einen Standardblock als Puffer, also 2048 Bytes.

=(STD,n) — vereinbart einen Pufferbereich von n Standardblöcken (PAM-Seiten). n darf maximal 16 sein (das sind 16 × 2048 Bytes = 32768 Bytes).

Ein-Ausgabe-Zustände

Gibt der Benutzer in der SELECT-Klausel "FILE STATUS IS datenname" an, so wird der Ein-Ausgabe-Zustand der zugehörigen Datei nach jeder Ein-Ausgabeoperation im Feld "datenname" abgelegt. Dieses Feld fragt der Benutzer ab, um Ein-Ausgabeoperationen zu kontrollieren. Die für eine indizierte Datei möglichen Werte sind in der folgenden Tabelle zusammengestellt.

Tabelle: Ein-Ausgabe-Zustände

FILE STATUS-Werte	Bedeutung
0×	erfolgreiche Ausführung
00	keine weitere Information
04	erfolgreicher READ, aber Satzlängenfehler (siehe COB1 Beschreibung [1]: RECORD-Klausel)
1×	erfolglose Ausführung: Endebedingung
10	READ bei Dateiende
16	READ nach bereits erkannter AT END-Bedingung
2×	erfolglose Ausführung: Schlüsselfehler
21	Beim Laden der Datei ist der zu schreibende Schlüssel entweder schon vorhanden oder entspricht nicht der aufsteigenden Reihenfolge.
22	WRITE für bereits vorhandenen Satz
23	READ auf einen nicht vorhandenen Datensatz (INVALID KEY-Bedingung)
24	unzureichende Sekundärzuweisung im FILE-Kommando unter Berücksichtigung von BLOCK DENSITY/PAD
3×	erfolglose Ausführung: permanenter Fehler
30	keine weitere Information
35	OPEN INPUT/I-O auf eine nicht vorhandene Datei; einer der folgenden Gründe liegt vor: 1. Das FILE-Kommando mit richtigem Linknamen fehlt; der Programmlauf wird mit der Meldung 9077 unterbrochen; bei OPEN INPUT/I-O wird das FILE-Kommando angefordert; 2. Datei ist nicht katalogisiert; 3. Datei ist zwar katalogisiert, aber OPEN und CLOSE wurden noch nicht ausgeführt.
38	OPEN auf eine Datei, die vorher mit CLOSE WITH LOCK geschlossen wurde
39	1. Im FILE-Kommando wurden einer oder mehrere der Operanden FCBTYPE, RECFORM, KEYPOS oder KEYLENGTH mit Werten angegeben, die von den entsprechenden expliziten oder impliziten Programmangaben abweichen. 2. Satzlängenfehler bei Eingabedateien (Katalogüberprüfung). 3. Satzlänge größer als BLKSIZE-Angabe im FILE-Kommando.
4×	erfolglose Ausführung: logischer Fehler
41	OPEN auf eine bereits eröffnete Datei
42	CLOSE auf eine nicht eröffnete Datei
43	DELETE oder REWRITE ohne vorherigen erfolgreichen READ bei ACCESS SEQUENTIAL
44	Überschreiten der Bereichsgrenzen: 1. WRITE oder REWRITE mit unzulässiger Satzlänge
46	sequentieller READ nach erfolglosem READ oder START
47	READ/START auf eine Datei, die nicht als INPUT oder I-O eröffnet wurde
48	Es wurde versucht, eine WRITE-Anweisung für eine Datei auszuführen, die nicht als OUTPUT oder I-O eröffnet ist.
49	DELETE oder REWRITE auf eine Datei, die nicht mit OPEN I-O eröffnet wurde
9×	sonstige erfolglose Ausführungen
90	Systemfehler; keine weitere Information
91	entweder Password-Fehler oder OPEN-Fehler
93	Fehler bei simultaner Aktualisierung (PAGE LOCK bei SHARUPD=YES)
94	1. Nur bei Simultanverarbeitung: Die Aufruffolge READ — REWRITE/DELETE wurde nicht eingehalten. 2. Die Satzlänge ist größer als die Blocklänge.

6.5.6 Simultanverarbeitung für Dateien mit indizierter oder relativer Organisation

6.5.6.1 Indizierte Dateien

Die Möglichkeit, eine ISAM-Datei für mehrere Benutzer gleichzeitig zugänglich zu machen, ist mit dem Operanden SHARUPD = YES im FILE-Kommando gegeben:

/FILE dateiname,LINK= . . . , SHARUPD = YES, . . .

Die folgende Tabelle zeigt, welche OPEN-Anweisungen für einen Benutzer B möglich sind, nachdem die Datei von Benutzer A bereits eröffnet wurde.

			Zulässige Angaben für Benutzer B					
		OPEN-Anweisung	SHARUPD = YES			SHARUPD = NO		
			INPUT	I-O	OUTPUT	INPUT	I-O	OUTPUT
von Benutzer A gewählte Angaben	SHARUPD = YES	INPUT	X	X		X		
		I-O	X	X				
		OUTPUT						
	SHARUPD = NO	INPUT	X			X		
		I-O						
		OUTPUT						

Die zulässigen Kombinationen von OPEN-Anweisung und SHARUPD-Angabe sind mit X gekennzeichnet.

Aus der Tabelle geht hervor, daß die Angabe von SHARUPD = YES für INPUT-Dateien überflüssig ist, falls alle Anwender OPEN INPUT verwenden. Wenn SHARUPD = YES für INPUT-Dateien trotzdem angegeben werden muß, da mindestens ein Anwender OPEN I-O verwendet, wird das nachfolgend beschriebene Sperren bzw. Entsperren nicht durchgeführt.

Die Angabe SHARUPD = YES im FILE-Kommando ist nur für die gleichzeitige Aktualisierung von einer oder mehreren ISAM-Dateien (OPEN I-O) durch zwei oder mehr Dialogbenutzer sinnvoll und auch notwendig.

Aktualisierungen im Stapelbetrieb sollten nacheinander ablaufen, um sowohl Logikfehler als auch Laufzeitverlängerungen zu vermeiden (unnötige Angabe von SHARUPD = YES kostet Laufzeit und CPU-Zeit).

Bei Angabe von SHARUPD = YES im FILE-Kommando wird automatisch auch WROUT = YES gesetzt, d. h. die ISAM-Puffer werden nach jeder Änderung sofort zurückgeschrieben. Dies ist aus Datensicherheits- und Eindeutigkeitsgründen erforderlich, erhöht aber wesentlich die Anzahl der Ein-Ausgaben.

Um Datenkonsistenz bei gleichzeitiger Aktualisierung einer indizierten Datei durch mehrere Benutzer zu gewährleisten, benutzt das COB1-Ablaufzeitsystem den Sperr- und Entsperrmechanismus der DVS-Zugriffsmethode ISAM. Dieser Mechanismus sorgt für das Sperren bzw. Entsperren der Datenblöcke, in denen die durch die Anweisungen READ, WRITE, REWRITE oder DELETE angesprochenen Datensätze liegen.

Datenblock ist das Vielfache einer PAM-Seite (= 2048 Bytes), das durch den BLKSIZE-Parameter im FILE-Kommando implizit oder explizit beim Erzeugen der Datei vereinbart wurde (siehe 6.5.6).

Ist im folgenden von Datensatzsperre die Rede, ist immer die Sperre des ganzen Blocks, in dem dieser Datensatz liegt, gemeint.

Für die Simultanverarbeitung von indizierten Dateien gibt es eine Formaterweiterung der READ- bzw. START-Anweisung, die jedoch nur wirksam wird, wenn im FILE-Kommando SHARUPD=YES angegeben und die Datei mit OPEN I-O eröffnet ist.

Formaterweiterung für alle Formate

READ dateiname [WITH NO LOCK] . . .
START dateiname [WITH NO LOCK] . . .

Regeln für die Simultanaktualisierung

1. READ- oder START-Anweisung mit WITH NO LOCK-Zusatz:

   Gibt Benutzer A WITH NO LOCK an und ist der entsprechende Datensatz vorhanden, wird dieser gelesen bzw. wird auf diesen positioniert, ungeachtet einer etwa durch einen anderen Benutzer bereits gesetzten Sperre. Der Datensatz wird nicht gesperrt. Eine REWRITE- bzw. DELETE-Anweisung kann auf einen so gelesenen Satz nicht ausgeführt werden.

   Ein simultaner Benutzer B kann denselben Datensatz sowohl lesen als auch aktualisieren.

2. READ- oder START-Anweisung ohne WITH NO LOCK-Zusatz:

   Gibt Benutzer A WITH NO LOCK nicht an und ist der entsprechende Datensatz vorhanden, kann eine READ- bzw. START-Anweisung nur dann erfolgreich ausgeführt werden, wenn der entsprechende Datensatz nicht bereits durch Benutzer B gesperrt ist. War die Ausführung der Anweisung erfolgreich, wird der Datensatz gesperrt. Vor Aufhebung der Sperre kann Benutzer B denselben oder irgendeinen anderen Datensatz desselben Datenblocks nur mit WITH NO LOCK lesen oder auf ihn positionieren, er kann aber keinen Datensatz dieses Datenblocks aktualisieren. (Hat Benutzer B die Datei mit OPEN INPUT eröffnet, kann er Sätze des gesperrten Datenblocks immer lesen.)

3. Aktualisierung von Datensätzen:

   Soll durch eine REWRITE- oder DELETE-Anweisung ein Datensatz aktualisiert werden, muß der betroffene Datensatz unmittelbar zuvor durch eine READ-Anweisung (ohne WITH NO LOCK-Zusatz!) gelesen werden. Nach dieser READ- und vor der REWRITE- bzw. DELETE-Anweisung darf für dieselbe indizierte Datei keine weitere Ein-Ausgabe-Anweisung ausgeführt werden. Zwischen diesen beiden Anweisungen dürfen für andere indizierte Dateien, deren FILE-Kommando SHARUPD=YES enthält und die zur gleichen Zeit mit OPEN I-O eröffnet sind, nur READ- oder START-Anweisungen mit WITH NO LOCK-Zusatz ausgeführt werden. Anweisungen für andere indizierte Dateien (ohne SHARUPD=YES und OPEN I-O) dürfen ausgeführt werden.

   Ein Verstoß gegen diese Vorschriften führt zu einer erfolglosen REWRITE- bzw. DELETE-Anweisung mit FILE STATUS 94.

4. Wartezeiten bei einer Sperre:

   Hat Benutzer A aufgrund einer erfolgreich ausgeführten READ- oder START-Anweisung einen Datensatz gesperrt und versucht Benutzer B auf denselben Datensatz oder irgendeinen anderen aus demselben Datenblock eine READ- oder START-Anweisung ohne WITH NO LOCK-Zusatz auszuführen, so führt dies für letzteren nicht sofort zum Mißerfolg. Benutzer B wird, falls er mit einem Betriebssystem BS2000 ab der Version 7.5 arbeitet, in eine Warteschlange eingeordnet, in der er auf die Freigabe der Sperre durch Benutzer A wartet. Erst wenn eine maximale Wartezeit abgelaufen und die Entsperrung innerhalb dieser Frist nicht erfolgt ist, gilt die Anweisung als erfolglos und FILE STATUS 93 wird gesetzt. Wurde die Sperre vor Ablauf der Wartezeit aufgehoben, so wird Benutzer B mit dem erfolgreichen Aufruf fortgesetzt.

   Falls Benutzer B ein Betriebssystem BS2000 in Version 7.1 und niedriger benutzt, werden intern maximal 99 Wiederholungen der Anweisung versucht; ist auch die letzte erfolglos, wird FILE STATUS 93 gesetzt.

5. Freigabe eines gesperrten Datensatzes:

   Ein Benutzer behält eine Datensatzsperre solange bei, bis er eine der folgenden Anweisungen ausführt:

   - erfolgreiche REWRITE- oder DELETE-Anweisung auf den gesperrten Datensatz
   - WRITE-Anweisung auf eine indizierte Datei, deren FILE-Kommando SHARUPD=YES enthält und die mit OPEN I-O eröffnet ist (d. h., auf dieselbe Datei, die den gesperrten Datensatz enthält, oder auf eine andere indizierte Datei; Entsperrung erfolgt auch bei Auftreten der INVALID KEY-Bedingung)
   - READ- oder START-Anweisung mit WITH NO LOCK-Zusatz auf dieselbe Datei (Entsperrung erfolgt auch bei Auftreten der AT END- oder INVALID KEY-Bedingung)
   - READ- oder START-Anweisung ohne WITH NO LOCK-Zusatz auf einen Datensatz innerhalb eines anderen Datenblocks derselben Datei (Entsperrung erfolgt auch bei Auftreten der AT END- oder INVALID KEY-Bedingung)
   - READ- oder START-Anweisung ohne WITH NO LOCK auf eine andere indizierte Datei, deren FILE-Kommando SHARUPD=YES enthält und die mit OPEN I-O eröffnet ist (Entsperrung erfolgt auch bei Auftreten der AT END- oder INVALID KEY-Bedingung)
   - CLOSE-Anweisung auf dieselbe Datei.

   Eine Anweisung für eine indizierte Datei kann also die Aufhebung einer Datensatzsperre auf einer anderen indizierten Datei bewirken.

Hinweise:

1. Soll in einem Programm eine indizierte Datei (SHARUPD=YES und OPEN I-O) bearbeitet werden, sollte für diese Datei eine „USE AFTER STANDARD ERROR"-Prozedur vorgesehen werden. In dieser Prozedur können die simultanverarbeitungsspezifischen FILE STATUS-Werte 93 (Datensatz von einem simultanen Benutzer gerade gesperrt) und 94 (REWRITE- oder DELETE-Anweisung ohne vorherige READ-Anweisung) abgefragt und angemessen verarbeitet werden.

2. Es sollte immer berücksichtigt werden, daß aufgrund des Block-Sperrmechanismus von ISAM beim Sperren eines Datensatzes auch zugleich alle Datensätze desselben Blocks für alle simultanen Benutzer gesperrt werden.
3. Für einen Benutzer kann höchstens ein Datenblock gesperrt, d.h., vor Aktualisierung durch andere Benutzer geschützt sein. Dies gilt auch dann, wenn er mehrere indizierte Dateien (alle SHARUPD = YES) im Modus OPEN I-O eröffnet hat.
4. Eine „Deadlock“-Situation (gegenseitiges Sperren von Datenblöcken durch verschiedene Benutzer) ist ausgeschlossen, da nur ein einziger Block aller ISAM-Dateien (alle SHARUPD = YES) gesperrt sein kann.
5. Falls zwischen einer READ- und einer REWRITE- bzw. DELETE-Anweisung für einen Datensatz ein Zugriff auf einen anderen Datenblock derselben oder einer anderen indizierten Datei erfolgt, der gleichzeitig eine Entsperrung des zuvor gesperrten Datenblocks zur Folge hat, muß der Datensatz vor Ausführung der REWRITE- bzw. DELETE-Anweisung noch einmal gelesen werden. Da der betroffene Datenblock in der Zwischenzeit für andere Benutzer entsperrt war, könnte der Inhalt des Datensatzes verändert worden sein (siehe Beispiel 46 a(a)).

   Erfolgt der Zugriff auf den anderen Datenblock bzw. die andere Datei ohne Sperrmechanismus, könnten die dadurch bereitgestellten Daten während der Verarbeitung bereits wieder durch simultane Benutzer verändert worden sein, ehe die REWRITE- bzw. DELETE-Anweisung ausgeführt worden ist (siehe Beispiel 46 a(b)).
6. Um zu vermeiden, daß ein Benutzer möglicherweise mit nicht mehr aktuellen Daten arbeitet, sollte der WITH NO LOCK-Zusatz nur dann verwendet werden, wenn dies unbedingt erforderlich ist.
7. Ein gesperrter Datensatz (Datenblock) führt bei simultanen Benutzern, die auf denselben Datensatz oder einen anderen desselben Blocks zugreifen wollen, zu Wartezeiten. Um diese so kurz wie möglich zu halten, sollte die Sperre sobald wie möglich wieder aufgehoben werden. Wird die Sperre nicht rechtzeitig aufgehoben, läuft die Wartezeit ab, und das Programm verzweigt in die vorgesehene USE-Prozedur, sofern vorhanden (siehe Beispiel 46 b) oder wird abgebrochen (mit Meldung 9067, FILE STATUS 93 und DVS-Fehlerschlüssel DAAA).

Beispiel 46 a: Lesen und Zurückschreiben in Datei ISAM1, wenn vor dem Zurückschreiben Daten aus einer Datei ISAM2 benötigt werden:

aa) Ohne WITH NO LOCK-Zusatz: zweimalige READ-Anweisung auf dieselbe Datei erforderlich, dafür Sperrzeiten kürzer:

Code	Erläuterung
`READ ISAM1 INTO WORK1` `    INVALID KEY . . .`	Lesen eines Datensatzes in ISAM1 und Zwischenspeichern in WORK1, betroffener Datenblock in ISAM1 gesperrt;
`READ ISAM2 INVALID KEY` . . . . Verarbeitung von WORK1 unter Berücksichtigung von ISAM2SATZ	Lesen eines Datensatzes in ISAM2, Aufhebung der Sperre in ISAM1, Sperrung des betroffenen Datenblocks in ISAM2;
`READ ISAM1 INVALID KEY` . . . . . Prüfung, ob ISAM1SATZ inzwischen geändert, gegebenenfalls erneute Verarbeitung	Erneutes Lesen des Datensatzes in ISAM1, Aufhebung der Sperre in ISAM2, Sperrung des betroffenen Datenblocks in ISAM1;
`REWRITE ISAM1SATZ FROM WORK1` `    INVALID KEY . . .`	Zurückschreiben des Datensatzes in ISAM1, Aufhebung der Sperre in ISAM1.

ab) Mit WITH NO LOCK-Zusatz: nur eine READ-Anweisung auf dieselbe Datei erforderlich, dafür Sperrzeiten länger:

Code	Erläuterung
`READ ISAM1 INVALID KEY . . .`	Lesen eines Datensatzes in ISAM1, betroffener Datenblock gesperrt;

`READ ISAM2 WITH NO LOCK`
`    INVALID KEY . . .`
.
.
.
.
.
Verarbeitung von ISAM1SATZ unter Berücksichtigung von ISAM2SATZ
.
.
.
.
.
	Lesen eines Datensatzes in ISAM2, betroffener Datenblock nicht gesperrt;

`REWRITE ISAM1SATZ INVALID KEY . . .`	Zurückschreiben des Satzes in ISAM1, Sperre wird aufgehoben.

Beispiel 46 b: Verzweigen zu einer „USE AFTER STANDARD ERROR"-Prozedur:

```
  .
  .
  .
  FILE-CONTROL.
      SELECT ISAM1
  .
  .
  .
      FILE STATUS IS FILESTAT1.
WORKING-STORAGE SECTION.
77  FILESTAT1    PIC 99.
  .
  .
  .
PROCEDURE DIVISION.
DECLARATIVES.
ISAM1ERR SECTION.
      USE AFTER STANDARD ERROR PROCEDURE ON ISAM1.
SPERRE.
      IF FILESTAT1=93
            DISPLAY „SATZ ZUR ZEIT GESPERRT" UPON TERMINAL,
            GO TO ISAM1ERR-EX.
FEHLERREST.
            DISPLAY „DMS-FEHLER ISAM1, FILE-STATUS=",
                     FILESTAT1 UPON TERMINAL.
ISAM1ERR-EX.                                         ──────►
    EXIT.
END DECLARATIVES.
STEUER.
  .
  .
  .
```

Verzweigen auf die Anweisung hinter der fehlerverursachenden Anweisung. Wie der aufgetretene Fehler sinnvoll zu behandeln ist, muß für den jeweiligen Anwendungsfall entschieden werden.

6.5.6.2 Relative Dateien

Dateien mit relativer Organisation können — ebenso wie indizierte — von mehreren Benutzern simultan aktualisiert werden, wenn das FILE-Kommando SHARUPD=YES enthält und die Datei mit OPEN I-O eröffnet ist.

Um Datenkonsistenz bei simultaner Aktualisierung zu ermöglichen, benutzt das COB1-Ablaufzeitsystem den Sperr- und Entsperrmechanismus der DVS-Zugriffsmethode UPAM. Die Zugriffskoordinierung erfolgt hier (anders als bei ISAM) dateispezifisch. Dies hat u. a. zur Folge, daß Anweisungen für eine Datei keine Auswirkungen für eine andere Datei haben.

Die Sperrung betrifft — wie bei ISAM — nicht einen einzelnen Datensatz, sondern den gesamten Datenblock, in dem sich der Datensatz befindet (siehe „Indizierte Dateien").

Wie für indizierte Dateien gibt es auch für relative Dateien (nur mit SHARUPD=YES, OPEN I-O) für alle Formate der READ- bzw. START-Anweisung die Erweiterung WITH NO LOCK (Beschreibung siehe „Indizierte Dateien").

Regeln für die Simultanaktualisierung

1. Das Lesen und Positionieren ohne bzw. mit WITH NO LOCK-Zusatz erfolgt wie bei indizierten Dateien.
2. Aktualisierung von Datensätzen:

   Soll durch eine REWRITE- bzw. DELETE-Anweisung ein Datensatz aktualisiert werden, muß der betroffene Datensatz (wie bei indizierten Dateien) unmittelbar zuvor durch eine READ-Anweisung (ohne WITH NO LOCK-Zusatz) gelesen werden. Zwischen beiden Anweisungen darf für dieselbe Datei keine weitere Anweisung ausgeführt werden. Anweisungen für andere relative Dateien sind jedoch — anders als bei indizierten Dateien — zulässig (aufgrund der dateispezifischen Zugriffskoordinierung).
3. Wartezeiten bei einer Sperre

   Die maximale Wartezeit auf die Freigabe eines gesperrten Blocks beträgt 999 Sekunden (Zeitbedingung, keine Wiederholungsschleife wie bei ISAM). Nach Ablauf dieser Zeit wird, falls vorhanden, die „USE AFTER STANDARD ERROR"-Prozedur angesprungen oder das Programm mit der Fehlermeldung 9067 beendet (FILE STATUS 93 und DVS-Fehlerschlüssel D9B0 oder D9B1).
4. Freigabe eines gesperrten Datensatzes

   Die Entsperrung eines gesperrten Datenblocks kann mit denselben Anweisungen bewirkt werden wie bei indizierten Dateien, jedoch müssen sich alle Anweisungen auf dieselbe Datei beziehen.

   Im Unterschied zu indizierten Dateien bewirkt also eine Anweisung für eine relative Datei keine Entsperrung von Datenblöcken einer anderen relativen Datei.

Hinweise

1. Soll in einem Programm eine relative Datei (mit SHARUPD=YES, OPEN I-O) verarbeitet werden, sollte für diese Datei eine „USE AFTER STANDARD ERROR"-Prozedur vereinbart werden (siehe „Indizierte Dateien").

2. Anders als bei indizierten Dateien (mit SHARUPD=YES, OPEN I-O) kann bei simultaner Verarbeitung mehrerer relativer Dateien (alle mit SHARUPD=YES, OPEN I-O) für jeden Benutzer je ein Datensatz in beliebig vielen Dateien gleichzeitig gesperrt (!) werden (innerhalb einer Datei immer nur 1 Satz). Dadurch kann es zu sogenannten „Deadlock"-Situationen kommen (siehe Beispiel 46c).

3. Wie bei indizierten Dateien sollte auch bei relativen Dateien die Sperre auf Datensätzen (Datenblöcken!) so schnell wie möglich aufgehoben werden, um die damit verbundenen Wartezeiten für andere Benutzer möglichst kurz zu halten.

Beispiel 46c: „Deadlock"

```
Benutzer A:                     Benutzer B:
READ datei1 (satz n)            READ datei2 (satz m)
  .                               .
  .                               .
READ datei2 (satz m)            READ datei1 (satz n)
(Block in Datei1                (Block in Datei 2
nicht entsperrt)                nicht entsperrt)
```

Beide Benutzer warten auf Freigabe des jeweiligen Blocks („Deadlock").

Die maximale Wartezeit auf die Freigabe eines gesperrten Blocks beträgt 999 Sekunden (Zeitbedingung, keine Wiederholungsschleife wie bei ISAM). Nach Ablauf dieser Zeit wird, falls vorhanden, die „USE AFTER STANDARD ERROR"-Prozedur angesprungen oder das Programm mit der Fehlermeldung 9067 beendet (FILE STATUS 93 und DVS-Fehlerschlüssel D9B0 oder D9B1).

6.6 Sortieren und Mischen

Mit Hilfe der SORT-Anweisung sortiert der Benutzer eine oder mehrere Dateien nach einer Reihe von Sortierschlüsseln. Mit Hilfe der MERGE-Anweisung mischt er mehrere sortierte Dateien zu einer Datei zusammen.

Zum Sortieren verwendet der COB1-Übersetzer die Sortierfunktion des BS2000 SORT [17]). Das Mischen wird vom COB1-System selbst durchgeführt.

Der COB1-Übersetzer legt im Objektprogramm einen Bereich an, in dem er die notwendigen Steueranweisungen für den Sortiervorgang des Dienstprogramms SORT bzw. für den Mischvorgang der COB1-Mischroutine generiert.

Die ausführliche Beschreibung der Anweisungen SORT und MERGE steht im Manual „COB1 Sprachbeschreibung", [1].

Dateien für das Sortierprogramm

Für einen Sortiervorgang werden folgende Dateien benötigt:

1. Sortierdatei:

   In dieser Datei (Arbeitsbereich) werden Datensätze sortiert. Nötig ist die Klausel

   SELECT sortierdatei ASSIGN TO DISC.

   Außerdem muß diese Datei in der Sortierdateierklärung (SD) der DATA DIVISION beschrieben sein. Mit der Anweisung

   SORT sortierdateiname . . .

   wird auf diese Datei zugegriffen.

   Ohne daß der Benutzer ein FILE-Kommando angibt, wird diese Datei unter dem Namen SORTWORK.Djjttt.TSnnnn (jj Jahr, ttt laufender Tag des Jahres, nnnn Prozeßfolgenummer) katalogisiert. Der Linkname ist SORTWK. Nach normalem Sortierende wird diese Datei gelöscht.

   Die Größe der Sortierdatei beim Einrichten ohne FILE-Kommando beträgt standardmäßig 24 x 16 = 384 PAM-Seiten. (Durch Versorgen von SORT-Sonderregistern kann dieser Wert beeinflußt werden.) Demnach ist die Primärzuweisung 384 PAM-Seiten. Die Sekundärzuweisung ist ¼ davon, also 96 PAM-Seiten.

   Mit dem Kommando

   /FILE dateiname,LINK=SORTWK,SPACE=(p,s)

   kann der Benutzer den Dateinamen und die Größe der Sortierdatei selbst bestimmen (siehe Manual „SORT", [17], Kapitel 4). Empfehlenswert ist dies bei großen Dateien. Nach normalem Sortierende wird diese Datei geschlossen, aber **nicht** gelöscht.

   SORT-Sonderregister (siehe Manual „COB1 Beschreibung", [1]):

   Vor dem Sortieren kann der Programmierer folgende SORT-Sonderregister versorgen:

   - SORT-FILE-SIZE: mit der Anzahl der Sätze.
   - SORT-MODE-SIZE: mit der durchschnittlichen Satzlänge.

     Diese beiden Register verwendet das Dienstprogramm SORT zur Berechnung der Dateigröße, d. h., der Programmierer kann indirekt den SPACE-Operanden beeinflussen.

   - SORT-CORE-SIZE: mit der gewünschten Größe der internen Arbeitsbereiche in Anzahl Byte.

     Durch diese Angaben kann der Programmablauf beeinflußt werden.

     Bei fehlender Angabe werden standardmäßig 24 * 4096 Byte, d. h. 24 Seiten zu je 4 K angenommen. Näheres siehe Manual SORT [17] Optimierung von Sortierläufen.

Nach SORT-, RELEASE- und RETURN-Anweisungen kann der Programmierer das SORT-Sonderregister

SORT-RETURN

abfragen: „0“ zeigt das ordnungsgemäße Sortieren an, „1“ das fehlerhafte Sortieren. Diese Abfrage empfiehlt sich, da bei fehlerhaftem Sortieren der Programmlauf nicht abgebrochen wird.

2. Eingabedatei(en):

   Ist keine Eingabeprozedur definiert, generiert der COB1-Übersetzer einen OPEN INPUT bzw. einen READ...AT END für die angegebene Datei. Jede Eingabedatei muß im COBOL-Programm definiert sein.

   Die Linknamen SORTIN und SORTINnn (01 ≤ nn ≤ 99) dürfen nicht innerhalb eines Sortierprogramms verwendet werden.

3. Ausgabedatei:

   Ist keine Ausgabeprozedur definiert, generiert der COB1-Übersetzer einen OPEN OUTPUT bzw. einen WRITE für die angegebene Datei. Die Ausgabedatei muß im COBOL-Programm definiert sein.

   Der Linkname SORTOUT darf nicht innerhalb eines Sortierprogramms verwendet werden.

Fixpunktausgabe für Sortierprogramme und Wiederanlauf

Die Angabe der RERUN-Klausel (Format 2) veranlaßt die Ausgabe spezieller Fixpunkte für Sortierdateien. Möglich ist auch die Angabe der APPLY CHECKPOINT-Klausel; sie wird intern in eine RERUN-Klausel aufgelöst. Fixpunkte enthalten Informationen über den Zustand des Sortiervorgangs. Sie sind notwendig, um ein vom Benutzer oder wegen Anlagenstörung abgebrochenes Programm wieder starten zu können, ohne den gesamten bisherigen Programmablauf zu wiederholen. Die Ausgabe von Sortier-Fixpunkten empfiehlt sich vor allem bei großen Mengen von zu sortierenden Daten, da auf diese Weise eine erfolgte Vorsortierung bei Programmabbruch nicht verloren geht.

Format 2 der RERUN-Klausel:

RERUN ON herstellername EVERY SORT OF sortierdateiname

herstellername: SYSnnn (000 ≤ nnn ≤ 200)

Fixpunkte werden in eine Fixpunktdatei (siehe 6.7) ausgegeben, die vom Sortierprogramm mit dem Standard-Dateinamen SORTCKPT.Djjttt.Tnnnn (jj Jahr, ttt laufender Tag des Jahres, nnnn tsn des laufenden Prozesses) und Standard-Linknamen SORTCKPT eingerichtet wird (siehe Manual „BS2000 SORT“ [17]). Mit dem SPACE-Operanden im Kommando

/FILE dateiname,LINK=SORTCKPT,SPACE=(p,s)

kann der Anwender die Größe dieser Datei selbst bestimmen. Die Fixpunktausgabe wird auf SYSOUT protokolliert (Meldung E301; siehe Abschnitt 6.7 und Manual „Systemmeldungen“, [18]). Den Zeitpunkt der Fixpunktausgabe kann der Anwender nicht selbst bestimmen.

Bei normaler Beendigung des Sortiervorgangs wird die Fixpunktdatei geschlossen, freigegeben und gelöscht, so daß der Benutzer keinen Zugriff auf sie hat.

Wird ein Sortierprogramm fehlerhaft abgebrochen, so kann man den Lauf beim zuletzt geschriebenen Fixpunkt wieder starten: Mit Hilfe der auf SYSOUT protokollierten Informationen gibt man dazu das RESTART-Kommando; siehe Abschnitt 6.7 und Manual „Kommandosprache“, [2].

Leere Seite seit Version 2.2A

6.7 Fixpunktausgabe und Wiederanlauf

6.7.1 Allgemeines

Fixpunkte werden von COB1 in eine externe Fixpunktdatei ausgegeben (ggf. zwei Fixpunktdateien, siehe unten). Ein Fixpunkt umfaßt Kennungsinformationen, Programmzustand, dazu bezogenen Systemzustand und virtuelle Speicherinhalte. Dies wird für einen möglichen späteren Wiederanlauf benötigt.

Durch Ausgabe solcher Fixpunkte kann ein absichtlich oder wegen Anlagenstörung abgebrochenes Programm zu beliebiger Zeit an der Stelle fortgesetzt werden, an der ein Fixpunkt ausgegeben wurde. Die Ausgabe von Fixpunkten empfiehlt sich vor allem bei Programmen mit langer Laufzeit, ist jedoch nur dann sinnvoll, wenn die Wiederherstellung der Ausgangsdaten bei einem eventuellen Wiederanlauf möglich ist.

6.7.2 Fixpunktausgabe

Die Ausgabe von Fixpunkten veranlaßt der Benutzer mit der RERUN-Klausel. Dabei kann er den Zeitpunkt der Fixpunktausgabe bestimmen; eine Ausgabe bei jedem Spulenwechsel für eine bestimmte Datei ist möglich sowie auch die Ausgabe nach Verarbeitung einer bestimmten Anzahl von Sätzen einer Datei.

Format 1 der RERUN-Klausel:

```
                                          [ {        { REEL }  }                ]
RERUN [ON herstellername] EVERY           { { END OF { UNIT }  } OF dateiname  } ...
                                          [ ganzzahl-1 RECORDS                  ]
```

herstellername Angabe SYSnnn (0 ≤ nnn ≤ 244)
COB1 erzeugt entweder eine Fixpunktdatei oder zwei Fixpunktdateien:

a) 1 Fixpunktdatei, falls nnn ≤ 200.
COB1 bildet den Standardnamen progid.RERUN.SYSnnn sowie den Linknamen SYSnnn.

b) 2 Fixpunktdateien, falls nnn >200.
COB1 bildet die Standardnamen progid.RERUN.SYS.nnnA, progid.RERUN.SYS.nnnB und die Linknamen SYSnnnA und SYSnnnB. Fixpunkte werden alternierend auf eine der beiden Fixpunktdateien ausgegeben.

Fixpunktausgabe in eine einzige Datei (SYS-Nr. ≤200): Fixpunkte werden fortlaufend geschrieben. Bei Dateiende wird intern weiterer Speicherbereich angefordert.

Fixpunktausgabe in zwei Dateien (SYS-Nr. >200): Fixpunkte werden alternierend in zwei Dateien geschrieben, wobei ein zuvor geschriebener Fixpunkt überschrieben wird.

Das Format 2 der RERUN-Klausel ist nur für Sortierdateien möglich und wird deshalb im Abschnitt 6.6 beschrieben.

Nach jeder fehlerfreien Ausgabe eines Fixpunktes werden dem Benutzer auf SYSOUT Informationen für einen eventuellen Wiederanlauf gemeldet.

Format der Meldung (siehe Manual „Systemmeldungen", [18]):

E301 CHECKPOINT # aa, HALF PAGE # =bb, DATE=cc, TIME=dd:ee

aa Fixpunkt-Nummer
bb PAM-Seiten-Nummer
cc mm/tt/jj: Monat/Tag/Jahr
dd Stunde
ee Minute

6.7.3 Wiederanlauf

Mit dem RESTART-Kommando startet der Benutzer ein ablauffähiges Programm bei einem durch einen Fixpunkt festgehaltenen Zustand.

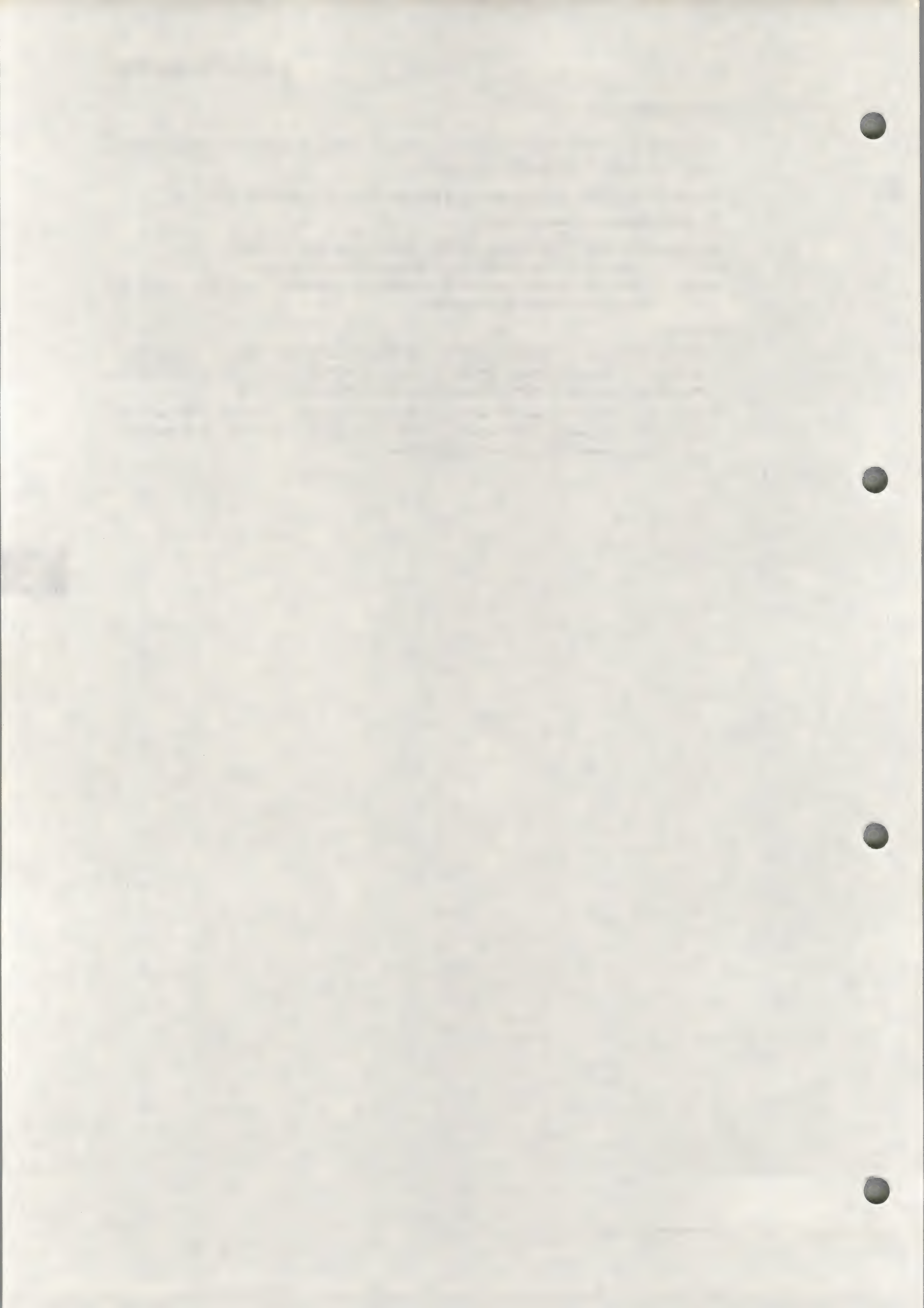
Format des RESTART-Kommandos (siehe Manual „Kommandosprache“, [2]):

RESTART dateiname,seite [,LOAD]

dateiname	Name der Fixpunktdatei (COBOL-Standardname; siehe 6.7.2)
seite	Nummer der PAM-Seite, in der die Fixpunktsätze beginnen
LOAD	Nach dem Laden wird das Programm nicht gestartet, sondern es wird in den Kommandomodus übergegangen.

Hinweise:

1. Für den Wiederanlauf muß der Benutzer alle Betriebsmittel zuweisen, die vom wiederanlaufenden Programm benötigt werden (FILE-Kommandos), da bei Ausgabe des Fixpunktes Angaben über benötigte Betriebsmittel nicht sichergestellt werden.
2. Der Zustand der Benutzerdaten wird beim Wiederanlauf **nicht** automatisch wiederhergestellt. Also muß der Benutzer selbst seine Daten so wie zum Zeitpunkt der Fixpunktausgabe in geeigneter Weise zur Verfügung stellen.



Anhang 1: Meldungen des COB1-Systems

Um die Diagnose von fehlerhaften COBOL-Programmen zu ermöglichen, wird vom COB1-Übersetzer und vom COB1-Ablaufzeitsystem eine umfassende Protokollierung aller Fehler vorgenommen. Dabei sind zwei Arten von Fehlerprotokollierung möglich:

1. Fehlermeldungen, die sich auf Fehler im zu übersetzenden COBOL-Programm beziehen und in einer Fehlerliste am Ende der Übersetzung ausgegeben werden.

   Diese Meldungen haben alle folgendes Format:

   Msg-Index Source Seq.No Severity Code Error Message

Msg-Index:	5-stellige Fehlerkennungsnummer, wobei die ersten beiden Ziffern das COB1-Übersetzersegment bezeichnen, das den Fehler erkannt hat.
Source Seq.No:	Zeilenfolgenummer der Zeile, in der der Fehler auftrat
Severity Code:	Angabe über die Fehlerklasse
Error Message:	Genauere Beschreibung des Fehlers; eventuell Umgehungsmöglichkeit (genauere Angaben sind der Beschreibung der Fehlermeldungsliste unter Punkt 2.3.4.4 zu entnehmen.)

   Nach der Überprüfung der Quelldateneingabe werden die Fehlermeldungen ausgegeben in der Folge

   — Identification Division
   — Environment Division
   — Data Division
   — Procedure Division

   und innerhalb dieser Ordnung nach aufsteigender Quellprogramm-Zeilennummer.

   Durch Angabe des COBRUN-Operanden ERDICT hat der Benutzer die Möglichkeit, sich jederzeit eine aktuelle Liste aller Fehlermeldungen zu erstellen, die vom COB1-Übersetzer erzeugt werden können. Beim Aufruf des COB1-Übersetzers und der Angabe COBRUN ERDICT ist kein Quellprogramm notwendig.

   Die Ausgabe der Fehlermeldungen erfolgt in englischer Sprache. Sollen die Fehlermeldungstexte in deutscher Sprache ausgegeben werden, so muß der COBRUN-Operand DIAGTEXT = GERMAN angegeben werden.

   Das folgende Beispiel zeigt eine Kommandofolge, mit der eine Liste aller COB1-Fehlermeldungen in deutscher Sprache erzeugt werden kann.

   Beispiel:

```
/PARAM CODE=2
/EXEC $COB1
 COBRUN ERDICT,DIAGTEXT=GERMAN
 END
```

Leerseite durch den Nachtrag vom September 1985

2. Fehlermeldungen, die durch Konflikte des COB1-Übersetzers bzw. des COB1-Ablaufzeitsystems mit der Systemumgebung auftreten. Jede Meldung enthält den Namen des Moduls, bei dessen Bearbeitung sie verursacht wurde.

   Diese Meldungen haben folgendes Format:

   $\left\{\begin{matrix}90\\91\end{matrix}\right\}$xx   Meldungstext

   Dabei bedeutet:

   $\left\{\begin{matrix}90\\91\end{matrix}\right\}$xx   4-stelliges Meldungskennzeichen

   Im gleichen Format werden auch einige Meldungen ausgegeben, die über Verlauf und Abschluß von Übersetzung und Anwenderprogrammablauf informieren.

Die Angabe des COBRUN-Operanden SSEQ#GEN bewirkt, daß die Meldungen 90xx und 91xx ergänzt werden mit der von COB1 vergebenen Quellprogramm-Zeilennummer der Anweisung, bei deren Ausführung die Meldung ausgegeben wurde.

A

Die folgende Liste enthält Meldungen des COB1-Übersetzers bzw. des Ablaufszeitsystems, und zwar:

- Meldungsnummer mit Meldungstext, englisch und deutsch; dies wird über SYSOUT ausgegeben;
- Erläuterungen, die für jede Meldung in gleicher Weise strukturiert sind (Typ — Bedeutung — Verhalten — Maßnahme).

9001 aaa TOTAL FLAGS: bbb / SI=ccc / S0=ddd / S1=eee / S2=fff / S3=ggg
9001 aaa FEHLER GESAMT: bbb / SI=ccc / S0=ddd / S1=eee / S2=fff / S3=ggg

Typ
Hinweis
Bedeutung
aaa: Programmname
bbb: Gesamtanzahl der Fehler
ccc: Anzahl der Severity-Code-I-Fehler
ddd: Anzahl der Severity-Code-0-Fehler
eee: Anzahl der Severity-Code-1-Fehler
fff: Anzahl der Severity-Code-2-Fehler
ggg: Anzahl der Severity-Code-3-Fehler

9002 COMPILATION OF aaa ABORTED
9002 DIE UEBERSETZUNG VON aaa WURDE ABGEBROCHEN

Typ
Anwenderfehler oder Systemfehler oder COB1-Fehler
Bedeutung
aaa: Programmname
Maßnahme
Fehler beheben, nochmal übersetzen; ggf. Systemberater verständigen

9003 LIBRARY aaa IS LOCKED. UNLOCK AND RESUME (R)
9003 DIE BIBLIOTHEK aaa IST GESPERRT. BITTE SPERRE AUFHEBEN UND RESUME (R) EINGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Bibliotheksname
Verhalten
Übersetzung unterbrochen
Maßnahme
Sperre aufheben und Kommando /R eingeben

9004 COMPILATION OF aaa USED bbb CPU SECONDS
9004 DIE UEBERSETZUNG VON aaa BENOETIGTE bbb CPU SEKUNDEN

Typ
Hinweis
Bedeutung
aaa: Programmname
bbb: Anzahl der Sekunden

9005 INCORRECT COBRUN/END CARD OR END CARD MISSING
9005 DIE COBRUN/END ANWEISUNG IST FALSCH ODER ES FEHLT DIE END ANWEISUNG

Typ
Anwenderfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
COBRUN- oder END-Anweisung korrigieren bzw. einfügen; nochmals übersetzen

9006 REASSIGNMENT OF SYSDTA NOT POSSIBLE
9006 EINE NEUZUWEISUNG VON SYSDTA IST NICHT MOEGLICH

Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9007 LIBRARY aaa HAS INVALID/NOT SUPPORTED TYPE. REASSIGN AND RESUME (R)
9007 DIE BIBLIOTHEK aaa HAT EINEN UNGUELTIGEN/NICHT UNTERSTUETZTEN TYP.
BITTE NEU ZUWEISEN UND (R) EINGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Bibliotheksname
Verhalten
Übersetzung unterbrochen
Maßnahme
richtige Bibliothek zuweisen und Kommando /R eingeben

9011 ERROR ON EXIT FROM THE USE-PROCEDURE ON PC: aaa
9011 FEHLER BEIM VERLASSEN DER USE-PROZEDUR AUF PC: aaa

Typ
Anwenderfehler
Bedeutung
aaa: Befehlszähler
Verhalten
Programm abgebrochen
Maßnahme
Programm korrigieren

9012 ATTEMPTING I/O OPERATION BEFORE INTERNAL FILE IS OPENED
9012 BEVOR EINE INTERNE DATEI GEOEFFNET WURDE WIRD VERSUCHT EINE EIN/AUSGABE OPERATION DURCHZUFUEHREN

Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahmen
Systemberater verständigen

9013 PARAMETER aaa NOT APPLICABLE TO THIS OPERATING SYSTEM
9013 DER OPERAND aaa IST FUER DAS BETREFFENDE BETRIEBSSYSTEM NICHT ANWENDBAR

Typ
Anwenderfehler
Bedeutung
aaa: COBRUN-Operand
Verhalten
Übersetzung läuft weiter; Operand wird übergangen

9015 ERROR OCCURRED LOADING OVERLAY aaa
9015 BEIM LADEN DES OVERLAYS aaa TRAT EIN FEHLER AUF

Typ
Systemfehler
Bedeutung
aaa: Nummer des Overlays
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9016 INSUFFICIENT SPACE FOR DYNAMIC ALLOCATION OF PAGES
9016 FUER DIE DYNAMISCHE SEITENZUWEISUNG IST NICHT GENUEGEND SPEICHERPLATZ VORHANDEN

Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9017 COMPILATION INITIATED; VERSION IS aaa
9017 BEGINN DER UEBERSETZUNG; VERSION aaa
Typ
Hinweis
Bedeutung
aaa: Versionsnummer des Compilers

9018 EAM WORKFILE/EAM * CANNOT BE OPENED
9018 DIE EAM-ARBEITSDATEI BZW. DIE EAM-DATEI * KANN NICHT GEOEFFNET WERDEN
Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9019 UNRECOVERABLE ERROR ON EAM *
9019 NICHT BEHEBBARER FEHLER AN DER EAM-DATEI *
Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9020 ERROR WHILE WRITING TO LIST-EAM OR SYSLST
9020 WAEHREND DES SCHREIBENS NACH LIST-EAM ODER NACH SYSLST TRAT EIN FEHLER AUF
Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9021 ERROR IN aaa MACRO CODE IS bbb
9021 FEHLER IN MAKRO aaa FEHLER CODE IST bbb
Typ
Systemfehler
Bedeutung
aaa: Name des Systemmakros
bbb: Inhalt von Register 15
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9022 aaa ERROR ON SAVLST FILE
9022 aaa FEHLER AN EINER SAVLST DATEI
Typ
Systemfehler
Bedeutung
aaa: DVS-Fehler-Code
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9023 UNRECOVERABLE EAM ERROR. PLEASE RECOMPILE
9023 ES TRAT EIN NICHT BEHEBBARER EAM FEHLER AUF. BITTE DEN UEBERSETZUNGSLAUF WIEDERHOLEN
Typ
Systemfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
nochmal übersetzen; ggf. Systemberater verständigen

9024 NOT ENOUGH MAIN MEMORY FOR COMPILATION. aaa MORE BYTES NEEDED
9024 FUER DIE UEBERSETZUNG IST NICHT GENUEGEND ARBEITSSPEICHERPLATZ ZUGEWIESEN. ES WERDEN NOCH aaa BYTES BENOETIGT

Typ
Systemfehler
Bedeutung
aaa: zusätzlicher Speicherbedarf
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9025 NOT ENOUGH MAIN MEMORY FOR GENERATION OF CROSS-REFERENCE (XREF) LISTING
9025 FUER DIE ERZEUGUNG DES QUERVERWEIS-PROTOKOLLS (XREF) IST NICHT GENUEGEND ARBEITSSPEICHERPLATZ ZUGEWIESEN! ES WERDEN NOCH aaa BYTES BENOETIGT

Typ
Systemfehler
Bedeutung
aaa: Anzahl der Bytes
Verhalten
Übersetzung abgebrochen
Maßnahme
ohne XREF wiederholen, ggf. Systemberater verständigen

9026 END OF PROCEDURE DIVISION OR ROOT SEGMENT ENCOUNTERED WITHOUT EXIT PROGRAM OR STOP RUN HAVING BEEN EXECUTED
9026 DAS ENDE DER PROCEDURE DIVISION BZW. DES ROOT-SEGMENTS WURDE ERREICHT OHNE DASS EXIT PROGRAM ODER STOP RUN AUSGEFUEHRT WURDE

Typ
Hinweis
Verhalten
Programmlauf wird abgebrochen
Maßnahme
An das logische Programmende eine STOP RUN-Anweisung setzen.

9027 SOURCE LINE TRUNCATED TO 80 CHARACTERS
9027 DIE QUELLPROGRAMMZEILE WURDE AUF 80 ZEICHEN VERKUERZT

Typ
Hinweis
Verhalten
Übersetzung läuft weiter
Maßnahme
ggf. Programm korrigieren

9029 LIBRARY aaa IS PROTECTED BY PASSWORD. PROVIDE PASSWORD AND RESUME (R)
9029 DIE BIBLIOTHEK aaa IST DURCH PASSWORT GESCHUETZT. BITTE (PASSWORD) UND (R) EINGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Bibliotheksname
Verhalten
Übersetzung unterbrochen
Maßnahme
Kommandos /PASSWORD und /RESUME eingeben

9030 LSD GENERATION SUPPRESSED BECAUSE OF SEVERITY 2 ERRORS
9030 DIE LSD GENERIERUNG WURDE WEGEN SEVERITY 2 FEHLERN UNTERDRUECKT

Typ
Hinweis
Verhalten
Übersetzung läuft weiter. Es werden jedoch keine LSD-Informationen für die Dialogtesthilfe AID erzeugt.
Maßnahme
Programm korrigieren

9031 LSD GENERATION FOR A SEGMENTED PROGRAM IS ONLY POSSIBLE WITH MODULE OUTPUT IN A PLAM LIBRARY

9031 DIE LSD GENERIERUNG FUER EIN SEGMENTIERTES PROGRAMM IST NUR BEI MODULAUSGABE IN EINE PLAM BIBLIOTHEK MOEGLICH

Typ
Hinweis
Verhalten
Übersetzung läuft weiter. Es werden jedoch keine LSD-Informationen für die Dialogtesthilfe AID erzeugt.
Maßnahme
Programm neu übersetzen und Bindemodul mit COBRUN MODULE = bibliotheksname in eine PLAM-Bibliothek ausgeben.

9032 LSD GENERATION SUPPRESSED BECAUSE OF INTERNAL I/O ERRORS

9032 DIE LSD GENERIERUNG WURDE WEGEN INTERNEN I/O FEHLERN UNTERDRUECKT

Typ
Systemfehler
Maßnahme
Systemberater verständigen

9033 LSD GENERATION SUPPRESSED BECAUSE OF INTERNAL ERRORS

9033 DIE LSD GENERIERUNG WURDE WEGEN INTERNEN FEHLERN UNTERDRUECKT

Typ
Systemfehler
Maßnahme
Systemberater verständigen

9036 LIBRARY aaa COULD NOT BE FOUND. REASSIGN AND RESUME (R)

9036 DIE BIBLIOTHEK aaa KONNTE NICHT GEFUNDEN WERDEN. BITTE NEU ZUWEISEN UND RESUME (R) EINGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Bibliotheksname
Verhalten
Übersetzung unterbrochen
Maßnahme
Kommandos /FILE . . . und /R eingeben

9038 COB1 RUN-TIME SYSTEM VERSION aaa IS INCOMPATIBLE WITH OBJECT CODE COMPILED BY COB1 VERSION bbb

9038 COB1-LAUFZEITSYSTEM VERSION aaa VERTRAEGT SICH NICHT MIT VON COB1 VERSION bbb UEBERSETZTEM OBJEKT-CODE

Typ
Anwenderfehler
Bedeutung
aaa: Versionsnummer
bbb: Versionsnummer
Das Laufzeitsystem ist älter als der Übersetzer, der das Programm erzeugte.
Verhalten
Programm abgebrochen
Maßnahme
Programm neu binden mit verträglichem RUN-TIME-System

9039 UNRECOVERABLE ERROR DURING DISPLAY UPON TERMINAL

9039 NICHT BEHEBBARER FEHLER WAEHREND EINER AUSGABE AUF TERMINAL

Typ
Systemfehler
Verhalten
Programm abgebrochen
Maßnahme
Systemberater verständigen

9040 ABNORMAL TERMINATION. USERS RETURN CODE=aaa. COBOL RETURN CODE=bbb
9040 ABNORMALE BEENDIGUNG. ANWENDER RETURN CODE=aaa. COBOL RETURN CODE=bbb

Typ
Anwenderfehler oder Systemfehler
Bedeutung
aaa: >0. Vom Programm wurde ein Anwender-Return-Code gesetzt, was zur Programmbeendigung führt.
bbb: >0. Vom COB1-System wurde ein Fehler festgestellt und zu Diagnosezwecken ein interner Return-Code gesetzt. Jedem Return-Code ist eine Fehlermeldung zugeordnet, aus der seine Bedeutung ersichtlich ist. Diese Fehlermeldung wird immer unmittelbar vor der Meldung 9040 ausgegeben. Falls das Programm von einer Jobvariable überwacht wird, wird außerdem der interne Return-Code in deren Rückkehrcode-Anzeige übernommen (siehe dazu Tabelle in Kap. 5.2).
Maßnahme
Programm bzw. Zuweisung ändern, ggf. Systemberater verständigen.

9044 UNALTERED GO TO
9044 GO TO OHNE ALTER

Typ
Anwenderfehler
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern

9045 ERRFIL FILE aaa CREATED AND CLOSED
9045 FEHLERDATEI aaa ERZEUGT UND GESCHLOSSEN

Typ
Hinweis
Bedeutung
aaa: Dateiname ERRFIL.COB1.programmname
Verhalten
Übersetzung läuft weiter

9046 aaa bbb ERROR ON ERRFIL FILE
9046 aaa bbb FEHLER BEIM ZUGRIFF AUF DIE FEHLERDATEI

Typ
Anwenderfehler oder Systemfehler
Bedeutung
aaa: DVS-Fehler-Code
bbb: einer der folgenden Texte: PASSWORD, LOCK, PARITY, NO SPACE, HARDWARE, OPEN, ALLOCATE, CATALOG
Verhalten
Übersetzung abgebrochen
Maßnahme
abhängig vom DVS-Fehler-Code: FILE-Kommando korrigieren;
bei Systemfehler Systemberater verständigen

9047 COMPILER ERROR. CLUMP LENGTH 0: OVERLAY aaa LAST SOURCE SEQUENCE NUMBER bbb LAST CLUMP TYPE ccc LAST CLUMP LENGTH ddd
9047 COMPILERFEHLER. CLUMP-LAENGE 0: OVERLAY aaa LETZTE QUELLPROGRAMMFOLGENUMMER bbb LETZTER CLUMPTYP ccc LETZTE CLUMPLAENGE ddd

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9049 aaa SWITCHES NOT SUPPORTED BY THIS OPERATING SYSTEM
9049 aaa SCHALTER WERDEN IN DIESEM BETRIEBS-SYSTEM NICHT UNTERSTUETZT

Typ
Anwenderfehler
Bedeutung
aaa: Name des Schalters
Verhalten
Schalter ignoriert, Programm läuft weiter
Maßnahme
ggf. Programm ändern

9050A AWAITING REPLY
9050A ANTWORT WIRD ERWARTET

Typ
Hinweis
Verhalten
Programmablauf unterbrochen
Maßnahme
Antwort eingeben für ACCEPT FROM CONSOLE

9051 END OF FILE ON ACCEPT FROM SYSIPT
9051 BEI ACCEPT FROM SYSIPT WURDE DATEIENDE ERKANNT

Typ
Hinweis
Verhalten
Programm läuft weiter
Maßnahme
Zuweisung von SYSIPT prüfen, ggf. neu zuweisen und Programm nochmal starten

9052 END OF FILE ON ACCEPT FROM SYSDTA
9052 BEI ACCEPT FROM SYSDTA WURDE DATEIENDE ERKANNT

Typ
Hinweis
Verhalten
Programm läuft weiter
Maßnahme
Zuweisung von SYSRDR prüfen, ggf. neu zuweisen und Programm nochmal starten

9053 UNRECOVERABLE ERROR ON ACCEPT FROM SYSDTA
9053 NICHT BEHEBBARER FEHLER BEI ACCEPT FROM SYSDTA

Typ
Systemfehler
Verhalten
Programm abgebrochen
Maßnahme
Systemberater verständigen

9054 ERROR OCCURRED TAKING CHECKPOINT, ERROR IS aaa
9054 BEIM SCHREIBEN EINES WIEDER-ANLAUFPUNKTES TRAT EIN FEHLER AUF, FEHLER IST aaa

Typ
Systemfehler
Bedeutung
aaa: einer der folgenden Werte in Register 15:
04 Arbeitsbereich im Klasse-5-Speicher konnte nicht erhalten werden.
08 Fehler in der Operandenliste des Benutzers.
0C Fixpunktdatei ist nicht eröffnet.
10 Keine Sekundärzuweisung oder ungültiges Schreiben.
14 FCB nicht PAM, oder OPEN nicht INOUT oder OUTIN.
18 Nicht-behebbaren Fehler vom DVS erhalten.
1C Fehler in der Katalogverwaltung
24 Fixpunkt konnte nicht gesetzt werden, weil ein gemeinsamer Speicherbereich verwendet wurde (Makro ENAMP ist wirksam).
28 Fixpunkt konnte nicht gesetzt werden, weil eine Serialisierungskennung vorhanden ist (Makro ENASI ist wirksam).
2C Fixpunkt konnte nicht gesetzt werden, weil eine Ereigniskennung vorhanden ist (Makro ENAEI ist wirksam).
30 Fixpunkt konnte nicht gesetzt werden, weil ein Contingency-Prozeß definiert wurde (Makro ENACO ist wirksam).
Verhalten
Programm abgebrochen
Maßnahme
Systemberater verständigen

9055 aaa EXCEPTION ON STATEMENT PC: bbb ccc STATUS=ddd eee
9055 aaa SONDERZUSTAND BEIM STATEMENT AUF PC: bbb ccc STATUS=ddd eee

Typ
Anwenderfehler bei der Fehlerbehandlung durch das Programm
Bedeutung
aaa: DML-/DVS-Fehler-Code
bbb: Befehlszähler
ccc: DB oder FILE
ddd: 5-stellige Zahl (bei DB) oder 2-stellige Zahl (bei FILE)
eee: bei FILE: DVS-Return-Code
Verhalten
Programm abgebrochen
Maßnahme
Programm prüfen, ggf. ändern

9056 SUB-SCHEMA MODULE TOO SMALL TO PROCESS AN EXTENSIVE DML-STATEMENT
9056 DER SUB-SCHEMA MODUL IST ZU KLEIN UM EIN UMFANGREICHES DML-STATEMENT ZU VERARBEITEN

Typ
Anwenderfehler (FIND-7, FETCH-7 ist zu umfangreich)
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern (kürzere DML-Anweisungen)

9057 NO CONNECTION WITH DATABASE DURING PROGRAM INITIALIZATION
9057 WAEHREND DER PROGRAMMINITIALISIERUNG KONNTE KEINE VERBINDUNG ZUR DATENBANK HERGESTELLT WERDEN

Typ
Anwenderfehler (entweder DBH nicht geladen oder FILE-Kommando inkorrekt)
Verhalten
Programm abgebrochen
Maßnahme
DBH laden und Programm mit evtl. geändertem FILE-Kommando neu starten

9058 UNRECOVERABLE ERROR DURING DISPLAY UPON SYSLST
9058 NICHT BEHEBBARER FEHLER BEI DISPLAY UPON SYSLST
Typ
Systemfehler
Verhalten
Programm abgebrochen
Maßnahme
Systemberater verständigen

9061 FORM-OVERFLOW WILL NOT OCCUR BECAUSE OF LACK OF SUPPORT IN BS2000
9061 UEBERLAUFBEDINGUNG WIRD NICHT AUFTRETEN, DA DIESE FUNKTION NICHT VOM BS2000 UNTERSTUETZT WIRD
Typ
Hinweis
Verhalten
Programm läuft weiter
Maßnahme
Programm ggf. ändern

9064 LOGGING WILL NOT BE PERFORMED BECAUSE OF LACK OF SUPPORT IN BS2000
9064 LOG-FUNKTION WIRD NICHT AUSGEFUEHRT DA IM BS2000 DIE SYSTEMUNTERSTUETZUNG FEHLT
Typ
Hinweis
Verhalten
Programm läuft weiter
Maßnahme
Programm ggf. ändern

9067 PROGRAM aaa PC bbb STATUS ccc FILE ddd DMS= eee NO USE ERROR PROCEDURE
9067 PROGRAMM aaa PC bbb STATUS ccc DATEI ddd DMS= eee KEINE USE PROZEDUR
Typ
Anwenderfehler oder Systemfehler: schwerwiegender Fehler bei Ein-Ausgabe oder Programmierfehler (keine USE-Prozedur, kein INVALID KEY oder keine AT END-Klausel)
Bedeutung
aaa: Programmname
bbb: aktueller Stand des Befehlszählers
ccc: aktueller COBOL FILE STATUS
ddd: Dateiname
eee: DVS-Fehlercode
Verhalten
Programm abgebrochen
Maßnahme
abhängig vom DVS-Fehler-Code: Programm, FILE-Kommando,... korrigieren oder bei Systemfehler Systemberater verständigen

9068 RPT SEQ# aaa: GROUP bbb REQUIRES TOO MANY LINES
9068 REPORT-QUELLPROGRAMMZEILEN# aaa: GRUPPE bbb ENTHAELT ZU VIELE ZEILEN
Typ
Anwenderfehler
Bedeutung
aaa: Zeilennummer
bbb: Name der Gruppe
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern

9069 RPT SEQ# aaa: GROUP bbb LINE CONFLICTS WIHT HEADING

9069 REPORT-QUELLPROGRAMMZEILEN# aaa: GRUPPE bbb EINE ZEILE STEHT IM WIDERSPRUCH ZUR SEITENKOPFBEGRENZUNG (HEADING)

Typ
Anwenderfehler
Bedeutung
aaa: Zeilennummer
bbb: Name der Gruppe
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern

9070 RPT SEQ# aaa REPEATED INITIATE

9070 REPORT-QUELLPROGRAMMZEILEN# aaa EINE INITIATE ANWEISUNG WURDE DURCHGEFUEHRT DOCH NICHT ABGESCHLOSSEN

Typ
Anwenderfehler
Der REPORT wurde nicht ordnungsgemäß mit einer TERMINATE-Anweisung abgeschlossen.
Bedeutung
aaa: Zeilennummer
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern

9071 RPT SEQ# aaa GENERATE ISSUED TO TERMINATED REPORT

9071 REPORT-QUELLPROGRAMMZEILEN# aaa FUER EINEN BEREITS ABGESCHLOSSENEN REPORT WURDE EINE GENERATE ANWEISUNG GEGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Zeilennummer
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern

9072A REPLY T (TERMINATE) OR D (DUMP)

9072A BITTE T (TERMINATE) ODER D (DUMP) EINGEBEN

Typ
Anwender- oder Systemfehler, der durch die vorhergehende Meldung beschrieben wurde.

9073 SORT NO. aaa UNSUCCESSFUL

9073 SORTLAUF MIT NR. aaa NICHT ERFOLGREICH

Typ
Anwenderfehler oder Systemfehler
Bedeutung
aaa: Nummer des SORT-Laufs
Verhalten
Programm läuft weiter
Maßnahme
Programm prüfen, ggf. SORT-RETURN abfragen

9074A VOLUME aaa UNEXPIRED PURGE DATE! REPLY "I" TO IGNORE

9074A FUER DEN DATENTRAEGER aaa IST DAS FREIGABE-DATUM NOCH NICHT ERREICHT. UM ZU IGNORIEREN "I" EINGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Archivnummer des Datenträgers
Verhalten
Programmlauf unterbrochen
Maßnahme
Falls Schutzmaßnahme ignoriert werden soll, "I" eingeben

9076 RPT SEQ# aaa: TERMINATE ISSUED TO REPORT WHICH IS NOT INITIATED
9076 REPORT-QUELLPROGRAMMZEILEN# aaa: EINE TERMINATE ANWEISUNG WURDE DURCHGEFUEHRT OBWOHL NOCH KEINE INITIATE ANWEISUNG GEGEBEN WURDE

Typ
Anwenderfehler
Es wurde versucht, einen nicht existierenden REPORT durch eine TERMINATE-Anweisung zu beenden.
Bedeutung
aaa: Zeilennummer
Verhalten
Programm abgebrochen
Maßnahme
Programm ändern

9077 LINK=aaa NO ENTRY IN CATALOG ISSUE NEW FILE COMMAND
9077 LINK=aaa KEIN KATALOG EINTRAG. BITTE NEUES FILE-KOMMANDO EINGEBEN

Typ
Anwenderfehler
Bedeutung
aaa: Linkname
Verhalten
Programm unterbrochen
Maßnahme
Gültiges FILE-Kommando und RESUME eingeben

9080A STOP LITERAL - AWAITING REPLY aaa
9080A STOP LITERAL - ANTWORT ERWARTET aaa

Typ
Hinweis
Bedeutung
aaa: ausgegebenes Literal
Verhalten
Programm unterbrochen; Meldung auf den Bedienungsplatz
Maßnahme
Operateur-Eingabe abwarten; Eingabe des Operateurs zur Programmfortsetzung ist beliebig

9081 THE DATABASE-HANDLER HAS NOT YET PROCESSED THE LAST DML-STATEMENT
9081 DER DATABASE-HANDLER HAT DAS LETZTE DML-STATEMENT NOCH NICHT ABGEARBEITET

Typ
Anwenderfehler
Der DBH ist durch STXIT unterbrochen und bekommt eine neue DML-Anweisung, ehe die unterbrechende abgearbeitet werden konnte.
Verhalten
Programm abgebrochen
Maßnahme
Programm korrigieren

9082 JOB-VARIABLES ARE NOT SUPPORTED IN THIS OPERATING SYSTEM
9082 JOB-VARIABLE WERDEN IN DIESEM BETRIEBSSYSTEM NICHT UNTERSTUETZT

Typ
Hinweis
Verhalten
Programm läuft weiter
Maßnahme
ggf. Programm ändern

9083 INTERNAL ERROR. THE ADDRESS CALCULATED BY GENERATION OF THE OBJECTLIST DOES NOT AGREE WITH THAT IN THE INTERNAL CLUMP
9083 INTERNER FEHLER. DER ADRESSPEGEL DER BEI DER ERZEUGUNG DER OBJEKTLISTE BERECHNET WURDE STIMMT NICHT MIT DEM ADRESSPEGEL IM CLUMP UEBEREIN

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9084 INTERNAL ERROR. THE ADDRESS CALCULATED BY GENERATION OF THE OBJECT MODULE DOES NOT AGREE WITH THAT IN THE INTERNAL CLUMP

9084 INTERNER FEHLER. DER ADRESSPEGEL DER BEI DER ERZEUGUNG DES OBJEKTMODULS BERECHNET WURDE STIMMT NICHT MIT DEM ADRESSPEGEL IM CLUMP UEBEREIN

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9085 INTERNAL ERROR. THE ADDRESS CALCULATED BY GENERATION OF THE OBJECT MODULE DOES NOT AGREE WITH THE START-ADDRESS OF AN LC SECTION IN THE CLUMP

9085 INTERNER FEHLER. DER ADRESSPEGEL DER BEI DER ERZEUGUNG DES OBJEKTMODULS BERECHNET WURDE STIMMT NICHT MIT DEM ADRESSPEGEL IM ORIGIN CLUMP AM BEGINN EINES NEUEN LC ABSCHNITTES UEBEREIN

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9086 INTERNAL ERROR. THE ADDRESS CALCULATED BY GENERATION OF THE OBJECT LIST DOES NOT AGREE WITH THE START-ADDRESS OF AN LC SECTION IN THE CLUMP

9086 INTERNER FEHLER. DER ADRESSPEGEL DER BEI DER ERZEUGUNG DER OBJEKTLISTE BERECHNET WURDE STIMMT NICHT MIT DEM ADRESSPEGEL IM CLUMP AM BEGINN EINES NEUEN LC ABSCHNITTES UEBEREIN

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9087 INTERNAL ERROR. A CLUMP WITH A LESS THAN ALLOWED LENGTH WAS READ

9087 INTERNER FEHLER. DIE MINIMALLAENGE EINES CLUMPS WURDE UNTERSCHRITTEN

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9088 ACCESS TO JOB-VARIABLE aaa FAILED. ERROR CODE=bbb

9088 FEHLERHAFTER ZUGRIFF ZUR JOB-VARIABLEN aaa FEHLER-CODE=bbb

Typ
Hinweis
Bedeutung
aaa: Linkname der JV
bbb: Code der Systemmeldung
Verhalten
Programm läuft weiter
Maßnahme
Programm ggf. ändern

9089 *STARTC NOT SUPPORTED IN THIS SYSTEM

9089 *STARTC WIRD IN DIESEM SYSTEM NICHT UNTERSTUETZT

Typ
Anwenderfehler (unzulässige Quellprogrammeingabe von Magnetband)
Verhalten
Übersetzung abgebrochen
Maßnahme
Bandinhalt in Plattendatei übertragen; neues FILE-Kommando; neu übersetzen

9090 RECORD DECLARATION STARTING ON LINE aaa TOO LARGE FOR COMPILATION
9090 DIE DATENSATZERKLAERUNG AB ZEILE aaa IST FUER DIE UEBERSETZUNG ZU GROSS

Typ
Hinweis
Bedeutung
Compiler-Einschränkung
aaa: Zeilennummer
Verhalten
Übersetzung abgebrochen
Maßnahme
Datensatzerklärung verkleinern

9091 ERROR DURING PLAM-ACCESS aaa
9091 FEHLER BEIM ZUGRIFF AUF PLAM-BIBLIOTHEK aaa

Typ
Systemfehler
Bedeutung
aaa: Name der PLAM-Bibliothek
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

9095 SAVLST FILE aaa CREATED AND CLOSED
9095 SAVLST DATEI aaa ERZEUGT UND GESCHLOSSEN

Typ
Hinweis
Bedeutung
aaa: Dateiname SAVLST.COB1.programmname
Verhalten
Übersetzung läuft weiter

9096 ERROR ON INTERFACE RUN-TIME-SYSTEM – OPERATING-SYSTEM IN ACCEPT OR DISPLAY STATEMENT
9096 FEHLER AN DER SCHNITTSTELLE LAUFZEITSYSTEM – BETRIEBSYSTEM IN ACCEPT ODER DISPLAY ANWEISUNG

Typ
Systemfehler
Verhalten
Programm abgebrochen
Maßnahme
Systemberater verständigen

9097 COMPILATION COMPLETED WITHOUT ERRORS
9097 DIE UEBERSETZUNG WURDE OHNE FEHLER BEENDET

Typ
Hinweis

9099 aaa
9099 aaa

Typ
Hinweis
Bedeutung
aaa: COBRUN-Anweisung
Verhalten
Übersetzung läuft weiter

9101 SUBSCRIPT-/INDEX-RANGE VIOLATION IN aaa STATEMENT IN LINE bbb, VALUE OF SUBSCRIPT/ INDEX IS ccc, TABLE BOUNDARY IS ddd THE PROGRAM eee

9101 UEBERSCHREITUNG DES SUBSKRIPT-/INDEXBEREICHS BEI ANWEISUNG: aaa IN SOURCEZEILE bbb, SUBSKRIPT- BZW. INDEXWERT: ccc TABELLENGRENZE: ddd PROGRAMM WIRD eee

Typ
Anwenderfehler
Bedeutung
aaa: Anweisung
bbb: Zeilennummer
ccc: Indexwert
ddd: maximaler Wert im OCCURS-Feld
eee: fortgesetzt/beendet
Verhalten
gemäß COBRUN RANGECHECK
Maßnahme
ggf. Programm ändern

9102 RANGE VIOLATION IN aaa STATEMENT IN LINE bbb, VALUE OF "DEPENDING ON" ELEMENT IS ccc, TABLE BOUNDARY IS ddd, THE PROGRAM ACTION eee

9102 UEBERSCHREITUNG DES SUBSKRIPT-/INDEXBEREICHS BEI ANWEISUNG aaa IN SOURCEZEILE bbb, WERT DES "DEPENDING ON"-ELEMENTS: ccc, TABELLENGRENZE: ddd, PROGRAMM WIRD eee

Typ
Anwenderfehler
Bedeutung
aaa: Anweisung
bbb: Zeilennummer
ccc: Wert im DEPENDING ON-Feld
ddd: maximaler Wert im OCCURS-Feld
eee: fortgesetzt/beendet
Verhalten
gemäß COBRUN RANGECHECK
Maßnahme
ggf. Programm ändern

9105 HARDWARE INTERRUPT ADDRESS: aaa OVERLAY: bbb SOURCE SEQUENCE NUMBER: ccc INTERRUPT WEIGHT CODE: ddd

9105 HARDWAREUNTERBRECHUNG BEI ADRESSE: aaa OVERLAY: bbb QUELLPROGRAMMFOLGENUMMER: ccc UNTERBRECHUNGSGEWICHT: ddd

Typ
Compilerfehler
Verhalten
Übersetzung abgebrochen
Maßnahme
Systemberater verständigen

Anhang 2: Aufbau des COB1-Systems

Das COB1-System besteht aus den Segmenten des Übersetzers und den Ablaufzeitmoduln.

Auf die Struktur des Übersetzers, die Namen der Segmente und die Funktion der notwendigen Ablaufzeitmoduln wird im folgenden näher eingegangen. Die Schaubilder sollen ein besseres Verständnis vom Übersetzungsvorgang und Ablauf der COBOL-Objektprogramme vermitteln.

Aufbau des COB1-Übersetzers

Der COB1-Übersetzer des Betriebssystems BS2000 besteht aus einer Anzahl von Segmenten, die in Overlay-Technik gebunden sind. Dabei übernimmt der Rootmodul die Kontrolle zum Nachladen übersetzungsabhängiger Überlagerungssegmente. Außerdem übernimmt der Rootmodul die Kommunikation einzelner Überlagerungssegmente untereinander und die gesamte Ein- und Ausgabe von COB1.

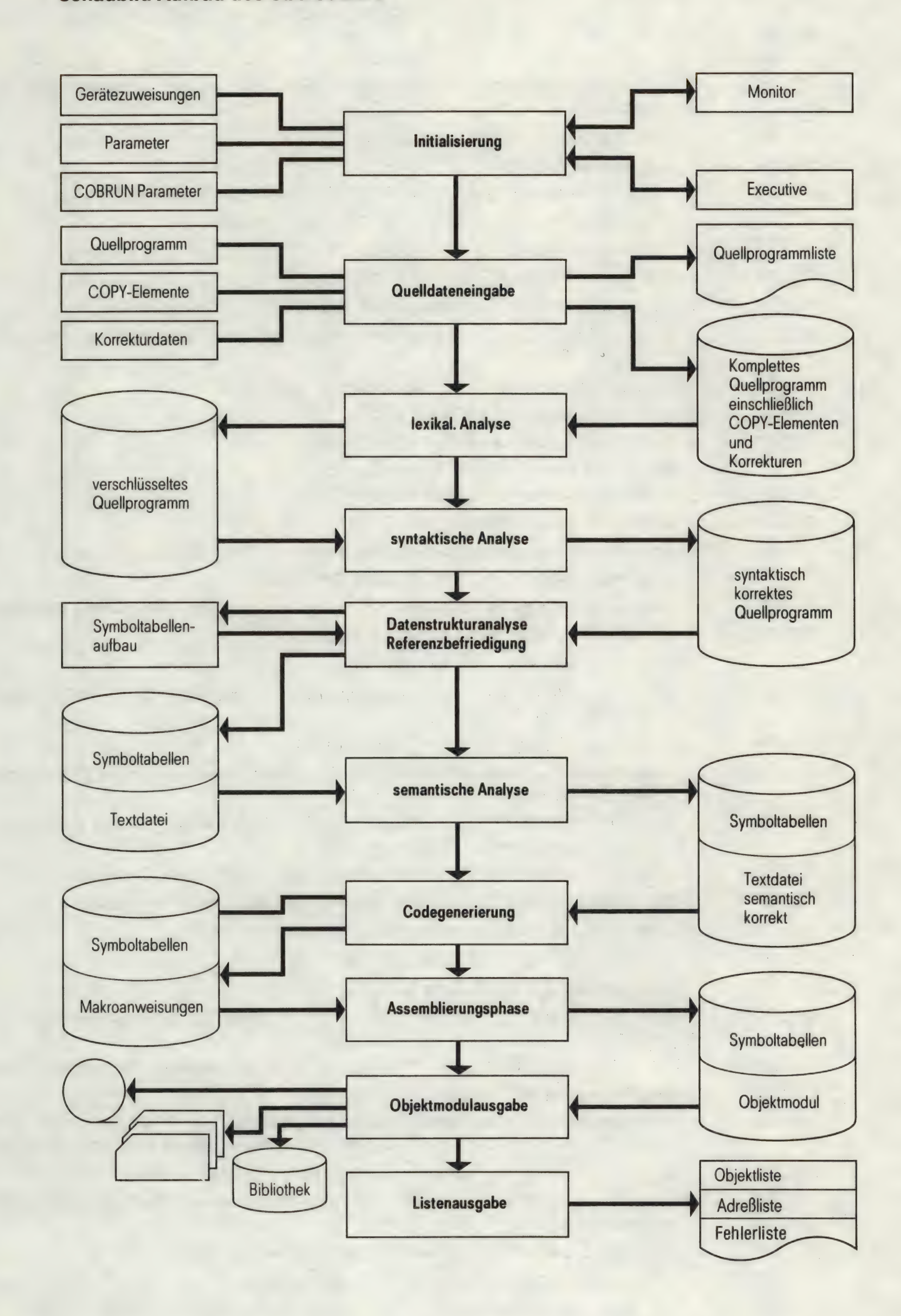
Die einzelnen Segmente bilden Funktionseinheiten, die durch den Ablauf einer COBOL-Übersetzung und durch die Struktur eines COBOL-Programms in die einzelnen DIVISIONS vorgegeben werden.

Man kann den Übersetzungsvorgang in folgende Funktionseinheiten gliedern:

1. Initialisierung
2. Quelldateneingabe
3. Lexikalische Analyse
4. Syntaktische Analyse
5. Semantische Analyse
6. Codegenerierung
7. Assemblierungslauf
8. Objektmodulgenerierung
9. Listenerzeugung

Der Aufbau des Übersetzers und die Anordnung der einzelnen Lademoduln im Arbeitsspeicher ist im folgenden Schaubild wiedergegeben.

Schaubild Aufbau des Übersetzers



Die Segmente des COB1-Übersetzers

Name	Funktion
ITCLØØ	Rootsegment; Koordinierungssegment, Schnittstelle für Arbeitsdateien, Systemschnittstelle
ITCLØ1	Initialisierung, Parameterübernahme
ITCLØ2	Eingabe der Quelldaten
ITCL1Ø	Koordinierungssegment für ITCL11, ITCL12, ITCL13, ITCL14
ITCL11	lexikalische Analyse für ID + ENVIRONMENT DIVISION
ITCL12	lexikalische Analyse für DATA DIVISION
ITCL13[2])	lexikalische Analyse für REPORT SECTION
ITCL14	lexikalische Analyse für PROCEDURE DIVISION
ITCL15[1])	Nachlauf für ITCL14 für segmentierte Programme zur Umordnung aufgespaltener Segmente.
ITCL21	syntaktische Analyse ID + ENVIRONMENT DIVISION
ITCL31	syntaktische Analyse DATA DIVISION
ITCL41	Datenstrukturanalyse für DATA DIVISION (Symboltabellenvervollständigung)
ITCL51[2])	syntaktische Analyse für REPORT SECTION
ITCL61[2])	Strukturanalyse für REPORT SECTION
ITCL7Ø[3])	syntaktische Analyse für DML-Anweisungen der PROCEDURE DIVISION
ITCL71	syntaktische Analyse für PROCEDURE DIVISION
ITCL81	Referenzbefriedigung für Daten und Prozedurnamen der PROCEDURE DIVISION
ITCLA1[4])	wie ITCL81 für CORRESPONDING-Datennamen
ITCL91[5])	Erzeugung der Querverweisinformation
ITCLBØ[3])	semantische Analyse der DML-Anweisungen der PROCEDURE DIVISION und erste Codegenerierung für DML
ITCLB1	Koordinierungssegment für ITCLBA, ITCLBB
ITCLBA[2])	semantische Analyse der REPORT WRITER Anweisungen und erste Codegenerierung für REPORT WRITER
ITCLBB	semantische Analyse der PROCEDURE DIVISION-Anweisungen und Zerlegung komplexer Source-Anweisungen in einfache.
ITCLB2	Analyse arithmetischer Ausdrücke, Verarbeitung der COBOL-Testverben
ITCLB3	Operandenanalyse + Generierung der PROCEDURE DIVISION-Literale
ITCLB4	Literal pooling, Tabellenvorbereitung für ITCLC1, ITCLD1
ITCLC1	Codegenerierung für PROCEDURE DIVISION
ITCLDØ[3])	Codegenerierung für DML-Anweisungen
ITCLD1	Codegenerierung für ENVIRONMENT + DATA DIVISION + Generierung der VALUE-Literale
ITCLD2	Umwandlung der Codegeneratorausgabe in System 4004/7.000 Maschinensprache
ITCLEA	Generierung der ISD-Sätze
ITCLE1	Verarbeitung und Generierung der Tabellen des generierten Objektmoduls (Adreßtabellen). Maschinencodegenerierung der Standardunterprogramme des Objektmoduls.
ITCLE2	Adreßauswertung und Abbildung auf explizite Speicheradressen des Objektmoduls.
ITCLE3	Generierung des Objektmoduls in TXT, ESD-Kartenformat und Erzeugung der Objektprogrammliste.
ITCLE4[6])	Sortieren des Adreßbuches in alphabetischer Reihenfolge.
ITCLF1[7])[5])	Erzeugung des Adreßbuches und der Querverweisliste
ITCLF2[8])[10])	Koordinierungssegment für ITCLF3, ITCLF4, ITCLF5, ITCLG3, ITCLG4, ITCLG5 und Fehlerlexikon 2. Stufe
ITCLF3[9])	Fehlerlexikon für englische Fehlermeldungen mit Kennung 02, 11, 12, 13, 14, 21, 31, 41
ITCLF4[9])[2])	Fehlerlexikon für englische Fehlermeldungen mit Kennung 51, 61 (REPORT WRITER)
ITCLF5[9])	Fehlerlexikon für englische Fehlermeldungen mit Kennung 7Ø, 71, 81, A1, BØ, B1, BA, BB, B2, B3, B4, C1, DØ, D1, D2, E1, E2
ITCLG3 ITCLG4[2])[9]) ITCLG5	entsprechend ITCLF . . für deutsche Fehlermeldungstexte

Die mit [1]) bis [10]) bezeichneten Übersetzermoduln werden nur in den unten näher beschriebenen Fällen verwendet:

[1]) Nur geladen, falls Segmente aus mehreren SECTIONS zusammengefügt werden müssen und dabei eine Umordnung des Quellprogramms notwendig ist.

[2]) REPORT SECTION innerhalb der DATA DIVISION.

[3]) SUB-SCHEMA SECTION innerhalb der DATA DIVISION (UDS).

[4]) CORRESPONDING-Angabe im Programm bei ADD, SUBTRACT, MOVE.

[5]) Angabe von PARAM XREF=YES.

[6]) Angabe von COBRUN MAPSRT oder MAPALL.

[7]) Angabe von PARAM MAP=YES.

[8]) Für englische Fehlermeldungstexte werden die ITCLF-Lademoduln verwendet.

[9]) Wenn Fehlermeldungen mit den angegebenen Kennungen vorliegen.

[10]) Für deutsche Fehlermeldungstexte werden die ITCLG-Lademoduln verwendet.

Das COB1-Ablaufzeitsystem

Das COB1-Ablaufzeitsystem liegt als Bibliothek von Moduln vor, die beim Binden eines COBOL-Objektprogramms zum ablauffähigen Programm verwendet wird. Das Hinzufügen der von COB1 ausgewählten Moduln wird automatisch vorgenommen. Die vorliegenden Moduln stellen COB1 bekannte Unterprogramme dar, die im wesentlichen in zwei Gruppen unterteilt werden können:

1. Unterprogramme für komplexe COBOL-Anweisungen

Beispiele für komplexe COBOL-Anweisungen sind Druckschrift [1] zu entnehmen (Beispiel SEARCH ALL...); aber auch für den Anwender scheinbar einfache Funktionen (wie z.B. COMPUTE A=B**C), für die keine entsprechenden Maschinenbefehle existieren, werden durch Bildung von Unterprogrammen und Auslagerung dieser Unterprogramme in vorübersetzte Moduln aufgelöst.

Die Namen und Funktionen dieser Unterprogramme sind aus der folgenden Tabelle zu entnehmen.

Name	Funktion
ITCØCLAØ ITCCCLAØ	Vergleich ALL ›Literal‹
ITCØCVBØ ITCCCVBØ	Umwandlung gepackt dezimal nach binär >15 Stellen
ITCØCVDØ ITCCCVDØ	Umwandlung binär nach gepackt dezimal >15 Stellen
ITCØCVFØ	Umwandlung von und nach Gleitpunkt
ITCØDPLØ	Division von Dezimalzahlen >15 Stellen
ITCØINIØ	INITIALIZE-Anweisung
ITCØINSØ	INSPECT-Anweisung
ITCØMDPØ	Berechnung variabler Adressen und Längen (rekursiv)
ITCØMPYØ ITCCMPYØ	Multiplikation von Dezimalzahlen >15 Stellen
ITCØMVEØ ITCCMVEØ	MOVE für alphanumerisch-druckaufbereitete Felder
ITCØRCHØ	Überprüfung Tabellengrenzen
ITCØREPØ	REPORT WRITER-Steuermodul
ITCØSCHØ ITCCSCHØ	SEARCH ALL-Anweisung
ITCØSTGØ	STRING-Anweisung
ITCCTCA1	Klassentest-Tabelle für Test auf ALPHABETIC
ITCCTCL1	Klassentest-Tabelle für Test auf ALPHABETIC-LOWER
ITCCTCP1	Klassentest-Tabelle für Test auf ALPHABETIC-UPPER
ITCCTCS1	Klassentest-Tabelle für Test auf NUMERIC (mit Vorzeichen)
ITCCTCU1	Klassentest-Tabelle für Test auf NUMERIC
ITCØTCVØ ITCCTCVØ	Klassentest bei Datenfeldern >256 Bytes oder Variablen

Name	Funktion
ITCØTRTØ ITCCTRTØ	TRANSFORM-Anweisung
ITCØUSTØ	UNSTRING-Anweisung
ITCØVCLØ	Vergleich für Felder variabler Länge/Adresse oder >256 Bytes
ITCØVMAØ	MOVE ALL ›Literal‹
ITCØVMPØ ITCCVMPØ	Auffüllen für Felder >256 Bytes bei MOVE
ITCØVMVØ	MOVE für Felder variabler Länge/Adresse oder >256 Bytes
ITCØXMNØ	EXAMINE-Anweisung
ITCØXPFØ ITCCXPFØ	Potenzierung mit Gleitpunktzahlen
ITCØXPIØ ITCCXPIØ	Potenzierung mit Ganzzahlen

2. Unterprogramme zum Anschluß des generierten Bindemoduls an Betriebssystemfunktionen

Diese Unterprogramme dienen hauptsächlich dazu, die Codegenerierung des Übersetzers möglichst betriebssystemunabhängig halten zu können. Die dabei möglicherweise auftretenden Effizienzverluste werden weitgehend durch die größere Betriebssystemunabhängigkeit wettgemacht. Bei Änderung der Schnittstellen genügt im allgemeinen das erneute Binden der vorhandenen Bindemoduln mit der neuen Ablaufzeit-Bibliothek (falls dies überhaupt notwendig sein sollte).

Wesentliche Funktionen unter diesem Titel sind:

. Anschluß der COBOL-Programme an das Ein-Ausgabesystem

. Anschluß der COBOL-Programme an SORT

. Anschluß der COBOL-Programme an UDS

. Anschluß der COBOL-Programme an Ablaufteil-Funktionen

Im folgenden sind die Namen und Funktionen der zu diesen Zwecken vorhandenen Ablaufzeit-Moduln näher beschrieben.

Name	Funktion
ITCØACAØ ITCCACAØ	ACCEPT-Anweisung
ITCØBEGØ	Programmsystem-Initialisierungsroutine
ITCMCKPØ	RERUN-Klausel mit Angabe ›Ganzzahl‹ RECORDS
ITCMCKP1	RERUN-Klausel für SORT-Dateien und END OF REEL
ITCØDBLØ	Verbindungsmoduln zum Datenbanksystem UDS
ITCMDCC1	physikalische Lese/Schreibroutine für direkte Dateien
ITCMDCLØ	CLOSE-Anweisung für direkte Dateien
ITCMDIN1	PAM-FCB-Generierung für direkte Dateien
ITCMDOPØ	OPEN-Anweisung für direkte Dateien
ITCMDRDØ	READ-Anweisung für direkte Dateien
ITCØDSAØ ITCCDSAØ	DISPLAY-Anweisung
ITCMDWRØ	WRITE/REWRITE-Anweisung für direkte Dateien
ITCØENDØ	Programmbeendigungsroutine (normal und abnormal)
ITCXERR1	Fehleranalyseroutine für Ein-Ausgabe
ITCØERTØ	ENTRY/RETURN-Anweisung
ITCCHSWØ	Setzen und Prüfen von Prozeß-/Benutzer-Schaltern
ITCMICLØ	CLOSE-Anweisung für indizierte Dateien
ITCMIIN1	ISAM-FCB-Generierung
ITCXINTØ	FCB-Initialisierung
ITCMIOPØ	OPEN-Anweisung für indizierte Dateien
ITCMIRDØ	READ/START-Anweisung für indizierte Dateien
ITCMIWRØ	WRITE/REWRITE-Anweisung für indizierte Dateien
ITCCMSG3 ITCDMSG3	Ausgabe von Fehlermeldungen

Name	Funktion
ITCØPCAØ ITCCPCAØ	Figurative Konstanten >256 Bytes bei COLLATING SEQUENCE
ITCØPCSØ ITCCPCSØ	Vergleiche für PROGRAM COLLATING SEQUENCE
ITCØPOVH	programmsystemspezifische Routinen und Konstanten
ITCMPUSØ	Versorgung des COBOL-Dateisteuerblocks mit Prozedurvereinbarungsadressen
ITCXRCC1	physikalische Lese/Schreibroutine für relative Dateien
ITCXRCLØ	CLOSE-Anweisung für relative Dateien
ITCXRIN1	PAM-FCB-Generierung für relative Dateien
ITCXROPØ	OPEN-Anweisung für relative Dateien
ITCXRRDØ	READ/START-Anweisung für relative Dateien
ITCXRWRØ	WRITE/REWRITE-Anweisung für relative Dateien
ITCMSCLØ	CLOSE-Anweisung für sequentielle Dateien
ITCØSEGØ ITCCSEGØ	Ansprung segmentierter COBOL-Programme
ITCØSFOØ ITCCSFOØ	APPLY-FORM-OVERFLOW-Routine für Druckerdateien (Leerfunktion im BS2000)
ITCMSIN1	SAM-FCB-Generierung
ITCMSLN1	LINAGE-Klausel bei WRITE für sequentielle Dateien
ITCØSMGØ	SORT/MERGE-Anweisung
ITCMSOPØ	OPEN-Anweisung für sequentielle Dateien
ITCØSPCØ	Druckersteuerzeichen-Auswertung
ITCMSRDØ	READ-Anweisung für sequentielle Dateien
ITCCST11	CODE SET-Tabelle für ASCII
ITCCST21	CODE SET-Tabelle für ISO-7
ITCØSTPØ ITCCSTPØ	STOP ›Literal‹-Anweisung
ITCMSWRØ	WRITE-Anweisung für sequentielle Dateien
ITCMULHØ	Benutzerkennsatzbehandlung
ITCØUPC2	Steuermodul für Prozedurvereinbarungen
ITCØUPS3	Sicherstellungsbereich für Prozedurvereinbarungen
ITCXXIT1	FILE STATUS und Fehlerbehandlungsroutine

Anhang 3: Beschreibung des Objektmodulformats

Struktur des Objektmoduls

Übersicht: Struktur des Objektmoduls

Kontrollabschnitt	Inhalt
Kontrollabschnitt (Ø)	Initialisierungscoding
Kontrollabschnitt (1)	• Standardunterprogramm für GO TO • Adreßkonstante für Tabellen • Indexnamenspeicher • Adreßkonstante für Datenbereiche • Adreßkonstante für Ablaufzeitroutinen • Adreßkonstante für FCB's/DTF's
Kontrollabschnitt (2)	• Adreßkonstante für angesprochene Paragraph- und Kapitelnamen • ALTER-Tabelle • Registersicherstellungsbereiche für PERFORM-Anweisungen • V-Adreßkonstante für Unterprogramme und Programmsegmente
Kontrollabschnitt (3)	• Steuerblöcke für Datenbanksprachelemente • Steuerblöcke für Ein- Ausgabesprachelemente • Arbeitsbereiche für spezielle COBOL-Anweisungen $\neq$ DML und I-O
Kontrollabschnitt (4)	• Speicherbereiche für Ein-Ausgabebereiche, die über temporäre oder permanente Register angesprochen werden (IOREG)
Kontrollabschnitt (5)	• Speicherbereiche für DATA DIVISION Datenerklärung und fest adressierte Ein-Ausgabebereiche • Arbeitsbereiche für Verben • Initialwerte von Daten (VALUE-Klausel)
Kontrollabschnitt (6)	• Programmspezifische Konstanten und COBOL-Register (TALLY, . . .) • intern verwendete Parameterbereiche zur Verfügung von Ablaufzeitroutinen • Adreßzeiger für Kontrollabschnitt (3) • Speicherbereiche für REPORT WRITER (LINE-COUNTER, SUM-COUNTER, . . .) • Speicherbereiche Konstanten der PROCEDURE DIVISION (Literale)
Kontrollabschnitt (7)	• Code für PROCEDURE DIVISION-Anweisungen • Code für übersetzerintern generierte Anweisungen (SORT, REPORT WRITER)
Kontrollabschnitt (8)	• Standardunterprogramme für GO TO, PERFORM, ALTER, EXIT für nichtsegmentierte Programme • Anschlußcode für segmentierte Programme zur GO TO, PERFORM, Behandlung • Programmsystemspezifische Konstante und Arbeitsbereiche } programm-unabhängig; standardmäßiger Zusatzmodul • Unterprogramme für komplexe Anweisungen der PROCEDURE DIVISION • Unterprogramme zur Bedienung der Betriebssystemschnittstelle • Speicherbereiche für Datenbank-kommunikationsbereich (UWA) } programm-abhängig nur, falls notwendig zur Bedienung bestimmter Sprachelemente }} zur Bindezeit automatisch aus Bibliothek geholt

Der Inhalt des Objektmoduls entspricht den allgemeinen Systemkonventionen. Er enthält ESD-, TXT-, RLD-Datensätze und im BS2000 gegebenenfalls zusätzlich ISD-Sätze. Eine genaue Beschreibung dieser einzelnen Bestandteile ist der Beschreibung [3] zu entnehmen.

Die **Anzahl der von COB1 erzeugten Objektmoduln** ist abhängig vom Aufbau des übersetzten Quellprogramms. Enthält das Quellprogramm keine SECTION-Nummern, oder keine DML-Anweisungen (siehe [9]), erzeugt COB1 nur einen Objektmodul.

Segmentierung

Werden SECTION-Nummern verwendet, die über der in der Klausel ›SEGMENT-LIMIT‹ angegebenen Grenze oder ≥50 liegen, werden von COB1 für aus diesen SECTIONS gebildeten Segmenten getrennte Moduln erzeugt.

Die Modulnamen werden aus dem PROGRAMM-ID-Eintrag gebildet. Dies ermöglicht entweder ein automatisches oder ein vom Benutzer gesteuertes Zusammenbinden dieser getrennt vorliegenden Moduln zu einem ablauffähigen Programm. Im BS2000 kann diese Funktion zur Bildung von ›shareable code‹-Segmenten unter Hinzunahme des DLL ausgenützt werden [3] (vgl. Abschnitt 6.2).

DML-Anweisungen

Wurde im COBOL-Quellprogramm eine SUB-SCHEMA SECTION angegeben, führt dies ebenfalls zur Bildung eines separaten Objektmoduls. Dieser Objektmodul dient als Platzhalter für die Aufnahme des Verbindungsbereiches zum Datenbanksystem (USER WORK AREA).

Namenskonventionen für den Objektmodul

In einem COBOL-Quellprogramm muß der Benutzer im PROGRAM-ID-Paragraphen das zu übersetzende Programm mit einem Namen versehen. Dieser Name wird vom Übersetzer als Grundlage zur Identifizierung des erzeugten Objektprogramms verwendet; d.h. ein Objektmodul muß in irgendeiner Form den im Quellprogramm angegebenen Namen enthalten. Der vom Benutzer im PROGRAM-ID-Paragraphen angegebene Name darf zu keinen Zweideutigkeiten mit anderen erzeugten Moduln oder im System vorhandenen Moduln (Ablaufzeitmoduln, die immer mit ITC beginnen) führen.

Da die erzeugten Objektmoduln (Bindemoduln) als Eingabe für den Binder dienen, dessen Hauptaufgabe darin besteht, beim Übersetzen noch nicht aufgelöste Adressen (d.h. externe Bezugnahmen) zu befriedigen, wozu der Name eines Objektmoduls dienen kann, muß der Programmname Bestandteil des ESD (External Symbol Dictionary) werden. Die im ESD angegebene Adresse ist dabei bei einem COBOL-Programm die Adresse des ersten zu durchlaufenden Befehls des Prozedurteils, d.h. die Anfangsadresse für den Ablauf des Programms. In Standard-COBOL stellt dieser Name die einzige Möglichkeit des Ansprungs eines COBOL-Programms dar. Dabei ist es gleichgültig, ob das Programm direkt vom Betriebssystem oder von einem Hauptprogramm aus angesteuert werden soll. Aus Kompatibilitätsgründen wird die Funktion der ENTRY-Anweisung weiterhin unterstützt. Sie entspricht jedoch nicht mehr der heutigen Erkenntnis der strukturierten Programmierung und sollte möglichst durch Verwendung von PROCEDURE DIVISION USING... ersetzt werden.

Es sind im folgenden zweierlei Arten von Namen zu unterscheiden:

1. Namen zum Aufruf eines Moduls. Diese Namen stellen externe Namen dar, die für Programmverknüpfungen beim Bindevorgang verwendet werden können.
2. Namen zur Benennung des Bibliothekseintrags, unter dem ein Modul in der Bibliothek gespeichert ist.

Wenn im folgenden von Namen gesprochen wird, sind damit immer die unter 1. angegebenen Namen gemeint.

Programme, die nur einen Modul erzeugen

Dazu gehören alle COBOL-Programme, die keine DML-Anweisungen enthalten, oder bei denen von der COBOL-Segmentierungsfunktion nicht Gebrauch gemacht wurde. Der im PROGRAM-ID-Paragraphen angegebene Name wird zur Kennzeichnung des Anfangspunktes des Programmablaufes verwendet.

Der Name einer Bibliothek ist entweder implizit festgelegt (bei der Bibliotheksdatei), oder er wird explizit durch einen LMR-Lauf bestimmt. Der Name eines Bindemoduls in der Datei „*" ist implizit durch den Namen des ersten ESD-Satzes gegeben.

Programme mit DML-Anweisungen

Für Programme, die eine SUB-SCHEMA SECTION enthalten, wird ein zusätzlicher Modul generiert, dessen Name aus dem SUB-SCHEMA-Namen gebildet wird. Mit dieser Maßnahme ist es möglich, verschiedene, getrennt kompilierte COBOL-Programme, die sich auf das gleiche SUB-SCHEMA beziehen, ohne Probleme miteinander zu einem Programmsystem zu verknüpfen, da nach der Kompilierung der Modul mit dem Namen des SUB-SCHEMA nur einmal zur Verfügung steht.

Externe Referenzen im Objektmodul

In diesem Abschnitt werden die im Abschnitt Objektprogrammliste erwähnten Teile aus der Sicht des Objektmoduls näher erläutert. Insbesondere wird der Unterschied zwischen externen Referenzen für Ablaufzeitroutinen, benutzerdefinierte Unterprogramme und Unterprogramm-Einsprungpunkte erklärt.

Ablaufzeitroutinen

Im Anhang 3 dieses Benutzerhandbuches sind die Namen und Funktionen der Ablaufzeitroutinen aufgeführt,die COB1 zur Ausführung komplexer Anweisungen auswählen kann. Im generierten Objektmodul erzeugt COB1 für jede ausgewählte Ablaufzeitroutine eine externe Referenz. Die Namen für diese externen Referenzen werden in den ESD-Sätzen des Objektmoduls abgesetzt. Für jede ausgewählte Routine erzeugt COB1 außerdem eine Adreßkonstante, die nach dem Binden des Programmes die Anfangsadresse der eingefügten Routinen enthält.

Anhand des in den ESD-Sätzen auftretenden Namens wird vom Binder ein Modul der COBOL-Ablaufzeit-Programmbibliothek gleichen Namens automatisch zum Objektmodul dazugebunden. Die Anfangsadresse dieses Moduls wird vom Binder in der entsprechenden Adreßkonstanten eingetragen. Dabei geschieht die Verknüpfung über die intern vergebenen ESID-Nummern. Zur Ablaufzeit verwendet das COBOL-Objektprogramm dann diese Adressen, um in das erforderliche Unterprogramm zu verzweigen.

Unterprogrammaufruf

Für jede CALL-Anweisung im COBOL-Quellprogramm wird von COB1 eine V-Konstante erzeugt. Diese ist über eine ESID-Nummer, die in einem ESD-Satz des Objektmoduls abgesetzt wurde, mit dem Namen des zu rufenden Unterprogrammes verknüpft. Bei Ausführung eines Unterprogrammaufrufs wird der Inhalt dieser V-Konstanten (der vom Binder eingetragen wurde) geladen und das entsprechende Unterprogramm angesprungen. Dies geschieht über einen vom Binder eingefügten Steuermodul. (Die Lage und der Name dieser V-Konstanten gehen ebenfalls aus der Beschreibung der Objektprogrammliste hervor). Näheres zur Steuerung dieser Unterprogrammaufrufe ist dem Abschnitt 6.3 zu entnehmen.

Unterprogramm-Einsprungpunkte

Externe Referenzen werden auch für Unterprogramm-Einsprungpunkte erzeugt. Diese können entweder direkt durch ENTRY-Anweisungen definiert sein oder implizit durch die USING-Angabe in der Anweisung PROCEDURE DIVISION vorhanden sein. Generell werden für Einsprungpunkte im COBOL-Objektprogramm Einträge in ESD-Sätzen mit dem Typ LD oder SD vorgenommen. Diese Namen sind der Beschreibung der Objektprogrammliste zu entnehmen. Beim Binden eines COBOL-Objektprogrammes werden diese Namen verknüpft mit entsprechenden Namen externer Referenzen in anderen dazugebundenen Moduln.

Anhang 4: Datenbankbedienung (UDS)

Eine Beschreibung des universellen Datenbanksystems UDS findet sich in den Manualen Entwerfen und Definieren [9], Aufbauen und Umstrukturieren [10], Anwendungen Programmieren [20], sowie im UDS Taschenbuch [19].

UDS-Datenbanken werden von Anwenderprogrammen bedient über

- COBOL-DML-Sprachelemente (DML ist integraler Bestandteil von COBOL)
- CALL DML (Datenbankbehandlung über Unterprogrammaufruf).

Der folgende Text beschränkt sich auf COBOL-DML. Ferner wird davon ausgegangen, daß Schema und Subschema bereits generiert sind. Hier werden einzelne Schritte zur Erzeugung eines UDS-Anwenderprogramms kurz dargestellt.

Der Database Handler (DBH) als Kernkomponente des UDS-Datenbanksystems ist zuständig für die Kommunikation zwischen dem Anwenderprogramm und der Datenbank (über das Subschema). Man unterscheidet:

- Linked-in DBH: Er wird in das Anwenderprogramm eingebunden, eignet sich also für den Fall, daß nur ein Anwenderprogramm mit der Datenbank arbeiten soll.
- independent DBH: Er wird nicht mit in das Anwenderprogramm eingebunden, d.h. er kann mehr als ein Anwenderprogramm steuern (eigener Prozeß).

Aufbau eines COBOL-DML-Programms

```
    .
    .
    .
DATA DIVISION.
    .
    .
    .
SUB-SCHEMA SECTION.
    DB subschema-name WITH IN schema-name

PROCEDURE DIVISION.
    Folge von COBOL-DML-Anweisungen
```

Die Formate der COBOL-DML-Anweisungen sind in [20] beschrieben.
schema-name/subschema-name: Werden bei der Schema- bzw. Subschemagenerierung festgelegt.

Übersetzen eines COBOL-DML-Programms

Der COB1-Übersetzer erzeugt aus einem COBOL-DML-Programm einen Programm-Modul und einen Subschema-Modul.

Mittels eines FILE-Kommandos (mit LINK=DATABASE) wird dem Übersetzer der Name der Datenbank (dbname) mitgeteilt. Dieser Name wurde schon bei der Datenbank-Generierung verwendet. Mit seiner Hilfe erkennt der Übersetzer die Datei dbname.COSSD, aus der er das Subschema kopiert. Sie wurde bei der Subschema-Generierung von UDS erzeugt.

Beispiel für eine Kommandofolge:

```
/ERASE *
/FILE dbname,LINK=DATABASE
/PARAM . . .
/SYSFILE SYSDTA=quellprogrammdatei
/EXEC $COB1
/SYSFILE SYSDTA=(SYSCMD)
```

Sicherstellen der von COB1 erzeugten Moduln

Wie und in welchen Bibliotheken die vom Compiler erzeugten Bindemoduln gesichert werden können, ist im Kapitel „Sicherstellung von Bindemoduln" beschrieben.

Binden eines COBOL-DML-Programms

Das Binden von COBOL-Programmen ist im Kapitel „Erzeugung ablauffähiger Programme" ausführlich beschrieben.

Bei COBOL-DML-Programmen ist jedoch zusätzlich zu beachten, daß je nach Wahl der DBH-Variante (=Database Handler) ein entsprechender UDS-Connection-Modul mit einzubinden ist (siehe hierzu [20]).

Beispiel eines Binderlaufs:

```
 /EXEC $TSOSLNK
 *PROG␣programmname[,FILENAM=dateiname]
 *INCLUDE␣cobol-dml-programm,modulbibliothek
 *INCLUDE␣uds-connection-modul,udsmodulbibliothek
[*RESOLVE␣,cobol-runtime-bibliothek]
```

Ablauf eines UDS-Anwenderprogramms

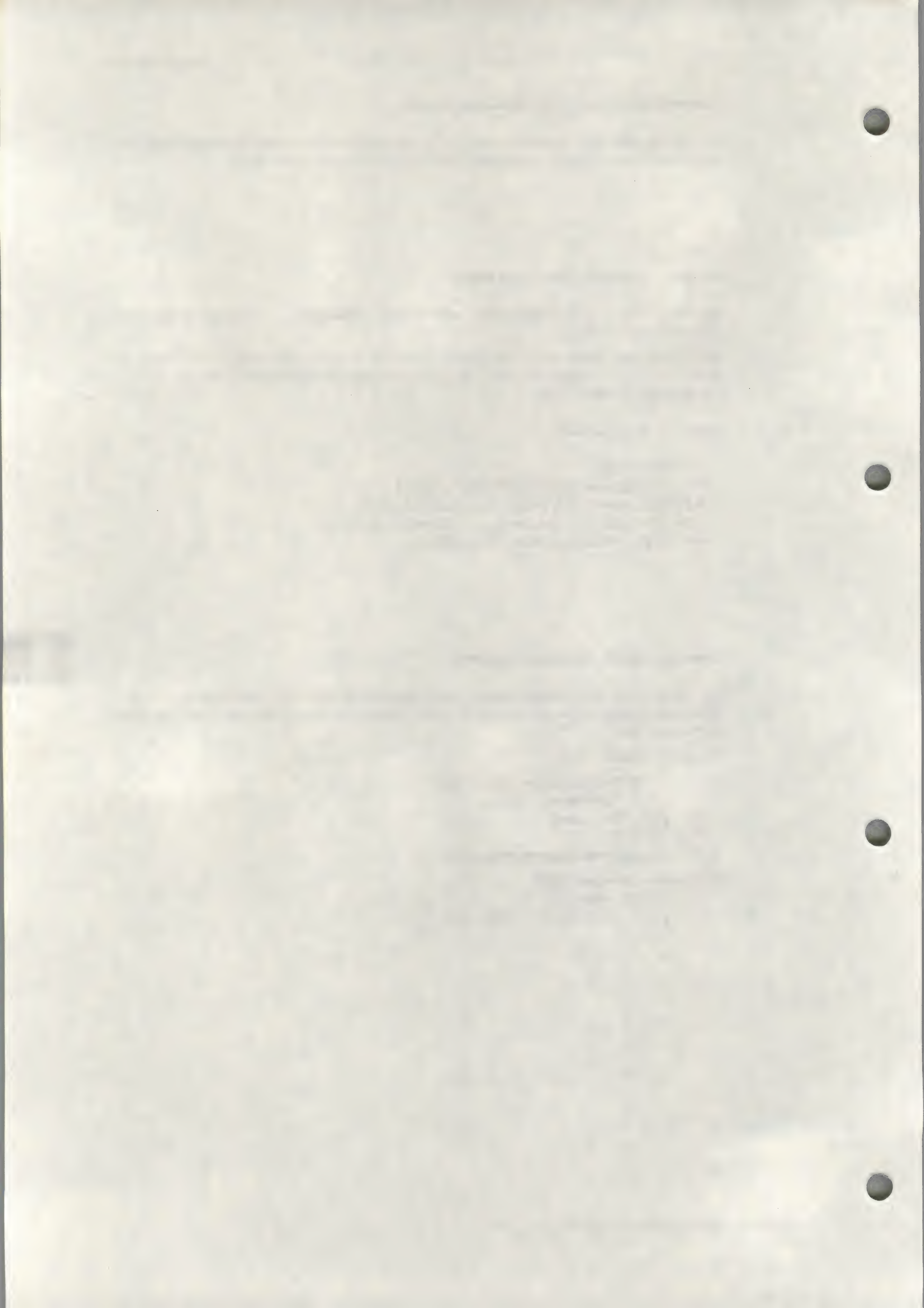
Der Ablauf eines UDS-Anwenderprogramms setzt bei Einsatz des independent DBH eine UDS-Session voraus. Die Verbindung zu dieser Session bzw. zur Datenbank stellt das FILE-Kommando her.

Ablauf mit linked-in DBH:

```
 /FILE dbname,LINK=DATABASE
 /EXEC dateiname
[DBH-Parameter]
 PP END
[Anwenderprogramm-Parameter]
```

Ablauf mit independent DBH:

```
 /EXEC dateiname
[Anwenderprogramm-Parameter]
```



Literatur

[1] **COB1** (BS2000)
COBOL-Compiler
Beschreibung

Zielgruppe
COBOL-Anwender im BS2000
Inhalt
Struktur und Elemente eines COBOL-Programms, Ein-/Ausgabebehandlung, Tabellenbearbeitung, Listenprogrammteil, Sortier- und Mischanschluß, Programmkommunikation, Segmentierung, Testhilfeelemente.

[2] BS2000
Kommandosprache des Organisationsprogramms
Beschreibung

Zielgruppe
BS2000 Anwender (nicht privilegiert)
Inhalt
Alle BS2000 Systemkommandos in lexikalischer Reihenfolge mit Hinweisen und Beispielen.
Folgende Liefereinheiten sind berücksichtigt:
BS2000-GA, MSCF, JV, FT, TIAM
Einsatz
BS2000 Dialogbetrieb, Prozeduren, Stapelbetrieb

[3] BS2000
Dienstprogramme
Beschreibung

Zielgruppe
BS2000 Anwender (nicht privilegiert)
Inhalt
Dienstprogramme für den nichtprivilegierten Benutzer des BS2000
Einsatz
BS2000 Teilnehmerbetrieb

[4] BS2000
DVS Plattenverarbeitung
Beschreibung

Zielgruppe
BS2000 Anwender (nicht privilegiert)
Inhalt
Funktionen des Datenverwaltungssystems im BS2000.
DVS-Kommandos und -Makroaufrufe, Service- und Aktionsmakroaufrufe.
Zugriffsmethoden UPAM, SAM, ISAM und EAM für Plattendateien.
Einsatz
BS2000 Dialogbetrieb, Stapelbetrieb, Programmierung

[5] **EDT** (BS2000)
Beschreibung

Zielgruppe
Datenerfasser, Programmierer
Inhalt
Beschreibung der Anweisungen an den Dateibearbeiter EDT, EDT-Prozeduren, Unterprogrammschnittstelle des EDT
Einsatz
BS2000 Dialog- und Stapelbetrieb

[7] BS2000
Dialog-Testhilfe
Beschreibung

Zielgruppe
Programmierer
Inhalt
Beschreibung der Kommandos und Makroaufrufe an die Dialogtesthilfe IDA.
Einsatz
BS2000 Dialogbetrieb

[8] BS2000
Dialog-Testhilfe
Kommandoformate

[9] **UDS** (BS2000)
Entwerfen und Definieren
Benutzerhandbuch

Zielgruppe
Datenbankentwerfer, Programmierer, Datenbankadministrator
Inhalt
UDS-Produktumfang, Grundzüge des Datenbank-Design, Datendefinitionssprache DDL, Speicherstruktursprache SSL, Subschema-Datendefinitionssprache Subschema-DDL.

[10] **UDS** (BS2000)
Aufbauen und Umstrukturieren
Benutzerhandbuch

Zielgruppe
Datenbankadministrator
Inhalt
Übersicht über die vom UDS benötigten Dateien,
UDS-Dienstprogramme, die zum Aufbauen der UDS-Datenbank nötig sind,
Dienstprogramme zum Umstrukturieren
Einsatz
Datenbankadministrator beim Aufbauen einer Datenbank

[11] TRANSDATA
UTM
Programmschnittstellen
(TRANSDATA BS2000)
Beschreibung

Zielgruppe
Programmierer, Organisatoren und Einsatzplaner
Inhalt
Einführung in UTM. Erläuterung des Programm-, Speicher- und Schnittstellenkonzepts, Beschreibung der Funktionsaufrufe sowie der Formaterstellung bei UTM, Informationen über den Anschluß von UTM an Datenbanken, Programmierhinweise.
Einsatz
BS2000 Transaktionsbetrieb

[13] BS2000
Systemverwaltung
Beschreibung

Zielgruppe
BS2000 Systemverwalter
Inhalt
Möglichkeiten und Aufgaben des Systemverwalters zur Steuerung und Verwaltung des Betriebssystems. Alle zu diesem Zweck benötigten Kommandos.
Einsatz
Systemverwaltung, Rechenzentrum

[14] TRANSDATA
UTM
Generierung und Administration
(TRANSDATA BS2000)
Benutzerhandbuch

Zielgruppe
Systemverwalter und Administratoren
Inhalt
Aufbau, Generierung und Betrieb von UTM-Anwendungen, Arbeiten mit UTM-Anwendungen, UTM-Testanwendung, UTM-Meldungen und Fehlercodes
Einsatz
BS2000 Transaktionsbetrieb

[15] BS2000
DVS Bandverarbeitung
Beschreibung

Zielgruppe
BS2000 Anwender, Assembler Programmierer (beide nicht privilegiert)
Inhalt
Funktionen des Datenverwaltungssystems im BS2000.
DVS-Kommandos und -Makroaufrufe, Service- und Aktionsmakroaufrufe.
Zugriffsmethoden UPAM, SAM und BTAM für Banddateien.
Einsatz
BS2000 Dialogbetrieb, Stapelbetrieb, Programmierung

[16] **FOR1** (BS2000)
FORTRAN-Compiler
Benutzerhandbuch

Zielgruppe
FORTRAN-Anwender im BS2000
Inhalt
Aufruf und Steuerung des FOR1-Compilers im BS2000, Eingabe und Übersetzung von Quellprogrammen, Verwaltung von Bindemoduln, Ablauf von FOR1-Programmen im BS2000, Testhilfen, Programmierhinweise, ferner Hinweise zu Sprachverknüpfungen sowie eine Auflistung der FOR1-Fehlermeldungen.

[17] **SORT** (BS2000)
Beschreibung

Zielgruppe
BS2000 Anwender
Inhalt
Funktionen und Anweisungen für das Sortieren und Mischen von Dateien

[18] BS2000
Systemmeldungen
Beschreibung

Zielgruppe
BS2000 Anwender
Inhalt
Standardmeldungen BS2000 V8.0 Zentralsystem, inklusive SPOOL, RSO V1.1, SDF V1.1.
Standardmeldungen der Softwareprodukte DCAM V8.1A, TIAM V8.1A, RBAM V8.1A

[19] **UDS** (BS2000)
Taschenbuch

Zielgruppe
UDS-Kenner, die bei der praktischen Arbeit schnell nachschlagen wollen
Inhalt
Zusammenstellung aller wichtigen Syntaxbeschreibungen, Tabellen und Entscheidungshilfen aus den UDS-Manualen.

[20] **UDS** (BS2000)
Anwendungen programmieren
Benutzerhandbuch

Zielgruppe
Programmierer
Inhalt
Transaktionskonzept, Funktionsweise der Currency-Tabelle, COBOL-DML, CALL-DML, Testen von DML-Funktionen

[21] **LMS** (BS2000)
Beschreibung

Zielgruppe
BS2000 Anwender
Inhalt
Beschreibung der Anweisungen zum Erstellen und Verwalten von Programmbibliotheken mit LMS
Einsatz
BS2000 Dialog- und Stapelbetrieb

[22] BS2000
Jobvariablen
Beschreibung

Zielgruppe
BS2000 Benutzer
Inhalt
Anwendungsmöglichkeiten für Jobvariablen zur Steuerung und Überwachung von Aufträgen und Programmläufen
Bedingungsabhängige Auftragssteuerung.
Alle erforderlichen Kommandos und Makroaufrufe.
Anwendungsbeispiele.
Einsatz
BS2000 Teilnehmerbetrieb

[23] BS1000
BS2000
TRANSDATA PDN
Systemkonventionen
Beschreibung

Zielgruppe
Benutzer von SIEMENS-Großrechenanlagen
Inhalt
Betriebssystemkonventionen für BS1000, BS2000 und TRANSDATA PDN, Konventionen für Datenträger, Codes für die Zeichendarstellung.

[24] BS2000
Binder und Lader
Beschreibung

Zielgruppe
BS2000 Anwender
Inhalt
Beschreibungen der Anweisungen zum Binden und Laden von Programmen mit TSOSLNK, ELDE und DLL
Einsatz
BS2000 Dialog- und Stapelbetrieb

[25] **AID** (BS2000)
Advanced Interactive Debugger
Testen von COBOL-Programmen
Benutzerhandbuch

Zielgruppe
COBOL-Programmierer
Inhalt
Vorbereitungen für das symbolische Testen von COBOL-Programmen. Beschreibung aller AID-Kommandos, die zum symbolischen Testen zur Verfügung stehen.
Beispiel einer AID-Sitzung.
Meldungen.
Einsatz
Testen von COBOL-Programmen im Dialog- und Stapelbetrieb.



Bestellung

Bitte bestellen Sie Siemens-Druckschriften "Datentechnik"

- als **Kunde** mit dem Bestellformular U1450-J-Z18-1. Sie erhalten es von Ihrem Ansprechpartner der zuständigen Zweigniederlassung bzw. Landesgesellschaft. Er wird Sie bei der Auswahl der Druckschriften gern unterstützen.
- als **Siemens-Mitarbeiter** mit dem
  - Inland Bestellzettel S2000 (blau), bzw.
  - Ausland Bestellzettel S2002 (weiß).

  Richten Sie Ihre Bestellung bitte an:

  ZVW LAGER
  Postfach 1500
  8510 Fürth

Bei der Bestellung geben Sie bitte die Bestellnummer vollständig an, damit der Ausgabestand der Druckschrift mit der bei Ihnen eingesetzten Produkt-Version übereinstimmt. Bestellnummern finden Sie in den Schriften:

Datentechnik
Druckschriftenverzeichnis
Bestell-Nr. U500

Datentechnik
Druckschriften-Neuerscheinungen

Das Druckschriftenverzeichnis erscheint zweimal jährlich und enthält die Bestelldaten aller verfügbaren Druckschriften aus dem Bereich Datentechnik.
Die Einzelblätter "Druckschriften-Neuerscheinungen" informieren wöchentlich über neue Druckschriften. Sie enthalten die Bestelldaten und eine kurze Inhaltsangabe. Zur Ablage dieser Blätter gibt es einen Ordner mit einem Register nach Sachgebieten. Er hat die Bestellnummer U1050-J-Z18-1 und kostet 8,60 DM.
Druckschriftenverzeichnis und "Druckschriften-Neuerscheinungen" erhalten Sie kostenlos und auf Wunsch regelmäßig. Als **Kunde** wenden Sie sich bitte an Ihren Ansprechpartner der zuständigen Zweigniederlassung bzw. Landesgesellschaft. Als **Siemens-Mitarbeiter** können Sie sich in den Verteiler aufnehmen lassen durch ein formloses Schreiben an:

SIEMENS AG, D ÖA, Otto-Hahn-Ring 6, 8000 München 83.

Änderungen von Manualen, die keine Neuausgabe erfordern, erscheinen als "Nachtrag" oder als kostenlose "Korrektur". Nachträge und Korrekturen werden ebenfalls über die "Druckschriften-Neuerscheinungen" angekündigt und müssen bei Bedarf gesondert bestellt werden.

Anstelle von Korrekturen wurden früher "Aktualisierungen" herausgegeben. Eine Aktualisierung müssen Sie nicht bestellen, wenn danach eine Korrektur oder ein Nachtrag erschienen ist. Die Aktualisierung ist in diesem Fall in die Korrektur bzw. den Nachtrag eingearbeitet.

Musterbestellungen finden Sie auf den folgenden Seiten.

Musterbestellung **Kunde**

Sie benötigen die Druckschrift
"COB1 Beschreibung" auf dem Stand der Produktversion 2.1.

Auszug aus dem Druckschriftenverzeichnis:

COB1	COBOL-Compiler, SW-Produkt BS1000, BS2000						
	Tabellenheft	(V1.11)	479	Ag	84	D 15/5774-01	1105
	Beschreibung	(V1.30)	978	Ag	492	D 15/5453-02	6970
	Nachtrag	(V1.30)	779	Ag	168	D 15/5453-03N1	3945
	Tabellenheft	(V1.11)	479	Ag	84	D 15/5774-01	1105
	Beschreibung	(V1.30)	978	Ag	492	D 15/5453-02	6970
	Nachtrag	(V1.30)	779	Ag	168	D 15/5453-03N1	3945
	Nachtrag	(V1.30)	1080	Ag	98	D 15/5453-04N2	1785
	Nachtrag	(V2.00)	1181	Ag	12	D 15/5453-05N3	0190
	Beschreibung	(V2.0)	482	Ag	666	U343-J-Z55-1	14880
	Aktualisierung	(V2.0)	682	Ag	1	U343-J1-Z55-2	0000
	Nachtrag	(V2.1)	383	Ag	277	U343-J2-Z55-3	3050
	Aktualisierung	(V2.1)	683	Ag	2	U343-J3-Z55-4	0000
	Korrektur	(V2.11)	184	Ag	50	U343-J4-Z55-5	+ 0000

Bestellung

über D-Druckschriften und -Formulare

Lieferanschrift:

Siemens AG
Zweigniederlassung
Vertrieb Datentechnik

Straße/Postfach

PLZ Ort

Firma

Bearbeiter

Straße

PLZ Ort

Bestell-Zeichen:

Hiermit bestellen wir aus dem Siemens-Druckschriftenverzeichnis Datentechnik zur Lieferung an oben genannte Anschrift:

Pos.	Bestell-Nr.	Kurztitel	Menge*	Einzelpr. DM	Gesamtpreis DM
1	U 343-J-Z55-1	COB 1 Beschreibung	2		
2	U 343-J2-Z55-3	COB 1 Beschreibung	2		
3	U 343-J3-Z55-4	COB 1 Beschreibung	2		
4					
5					

Ort, Datum

Bestell-Nr. U1450-J-Z18-1

Firmenstempel und Unterschrift

Musterbestellung **SIEMENS-Mitarbeiter (Inland)**

Sie benötigen die Druckschrift
"Generierung eines Datenkommunikationssystems" für BS2000 V7.1, PDN V7.0.

Auszug aus dem Druckschriftenverzeichnis:

Generierung eines Datenkommunikationssystems					
Benutzerhandbuch (BS1000 V1.6, BS2000 V6.0, PDN V6.0)	182	Ag	266	U519-J-Z75-1	10100
Benutzerhandbuch (BS1000 V1.6, BS2000 V6.0, PDN V7.0)	382	Ag	302	U519-J-Z75-2	6830
Nachtrag (BS1000 V1.6I, BS2000 V7.1, PDN V7.0)	1082	Ag	156	U519-J1-Z75-3	1760

Bitte ergänzen, Nichtzutreffendes streichen

An

ZVW Lager · 8510 Fürth · Postfach 1500
~~ZVW Foto-Atelier Erlangen~~
~~ZVW~~ ....

Bestellzettel

Nummer

Dienststelle / Besteller

Unternehmensbereich

Bearbeiter

Datum

Liefertermin

Unterschrift

Versandanschrift

Versandart

ZVW-Arbeitsnummer (wird von ZVW belegt)

Zusätzlich zu Kostenstelle u. Kostenart ist unbedingt der Rechnungsempfänger anzugeben (s. Ausfüllhinweise)

Rechnungsempfänger	Art	Auftragskennzeichen: Kostenstelle	Kostenart
	G	9	

Positions-Nr.	Bestell-Nr.* Bezeichnung der Leistung	Menge	Einheit**	Einzelpreis DM	Gesamtpreis DM
1	U 519-J-Z75-2 Generierung eines Datenkomm.syst.	1			
2	U 519-J1-Z75-3 - " -	1			

K 3037 (S 2000) 9.83 Mat.-Nr.: 004 30 397

Musterbestellung **SIEMENS-Mitarbeiter (Ausland)**

Sie benötigen die Druckschriften
"Control System Command Language" und "Executive Macros" für BS2000 7.1.

Auszug aus dem Druckschriftenverzeichnis:

Control System Command Language						
Reference Manual (V5.0)	579	Ag	128	D 15/5136-03-101	2720
Revision (V5.0)	979	Ag	80	D 15/5136-04N1-101	0750
Revision (V5.1)	180	Ag	120	D 15/5136-05N2-101	1020
Reference Manual (V6.0)	980	Ag	376	D 15/5136-06-101	3740
Revision (V6.2)	381	Ag	126	D 15/5136-07N1-101	1190
Reference Manual (V7.1)	982	Ag	480	U808-J-Z55-1-7600	5200
Update (V7.1)	583	Ag	9	U808-J1-Z55-2-7600	0000

Executive Macros					
Reference Manual (V5.0)	479	Ag	480	D 15/5135-03-101	4590
Revision (V5.0)	979	Ag	88	D 15/5135-04N1-101	1240
Reference Manual (V5.0/V5.1)	480	Ag	528	D 15/5135-05-101	4000
Reference Manual (V6.0)	680	Ag	600	D 15/5135-06-101	6200
Revision (V6.2)	181	Ag	62	D 15/5135-07N1-101	1105
Reference Manual (V7.1)	1082	Ag	801	U810-J-Z55-1-7600	6370
Update (V7.1)	583	Ag	6	U810-J1-Z55-2-7600	0000

Nichtzutreffendes streichen / delete where not applicable / Tachar lo que no corresponde

An
Siemens Aktiengesellschaft
~~ZVW 16 · Postfach 103 · D-8000 München 1~~
ZVW 85 · Postfach 1500 · D-8510 Fürth-Bislohe

Bestellzettel

Nummer/Number/Número

Besteller/Ordered by/Pedido por

Zeichen/Reference/Referencia

Datum/Date/Fecha

Liefertermin/Delivery date/Plazo de suministro

Unterschrift/Signature/Firma

Versandanschrift/Forwarding address/Dirección del destinatario

Versandart/Method of dispatch/Forma de envio

ZVW-Arbeitsnummer (wird von ZVW belegt/please leave blank/favor dejar libre)

Auftragskennz. 1 unbedingt angeben / **Reference 1** is obligatory / **Referencia 1** imprescindible

Auftragskennz. 1/Reference 1/Referencia 1: Rechnungsempf. | Art | GBK | Lfd. Nummer

Auftragskennz. 2/Reference 2/Referencia 2

Positions-Nr. Item Pos. No.	Bestell-Nr.*/Bezeichnung der Leistung Order No.*/Description of service No. de pedido*/Descripción del servicio	Menge Quantity Cantidad	Einheit Unit Unidad **)	Einzelpreis Unit price Precio unitario DM	Gesamtpreis Total price Precio total DM
1	U 808-J-Z55-1-7600 Command Language	2			
2	U 808-J1-Z55-2-7600 Command Language	2			
3	U 810-J-Z55-1-7600 Executive Macros	1			
4	U 810-J1-Z55-2-7600 Executive Macros	1			

Stichwörter

Ablauf der Übersetzung 2-57
Ablauffähiges Programm 2-1, 5-1, 5-9
Ablaufzeitmoduln A-19
Ablauf (Programm) 5-5, 5-8, 6-27, 6-35, 6-40, 6-47
Ablaufzeitroutine A-19
Ablaufzeitsteuerung 6-27, 6-35, 6-40
Ablaufzeitsystem 4-1, 6-12, A-19
Abspeichern auf Magnetband 1-4
ACCEPT-Anweisung 5-1, 6-21
ACCESS-Klausel 6-24ff, 6-27, 6-37, 6-44
ACTKEY (COBRUN-Operand) 2-14, 2-57, 2-62
Adreßliste 2-17, 2-19, 2-24, 2-50
Änderungen
— in Dateien 1-8
— in LMS-Bibliotheken 1-21
— mit EDT 1-9
AID (Dialogtesthilfe) 5-11
Aktualisierung
— von indizierten Dateien 6-49ff
— von relativen Dateien 6-56ff
APPLY BLOCK DENSITY-Klausel 6-26
ASSIGN-Klausel 5-7
Aufbau
— COB1-System A-16
— Datenblock 6-45ff
— indiziert organisierte Datei 6-44
— relativ organisierte Datei 6-37
— sequentiell organisierte Datei 6-27
— Übersetzer A-16
Aufgerufenes Assembler-Programm 6-16
Aufrufendes Assembler-Programm 6-19
Ausgabe
— Bindemoduln 2-22
— des Übersetzers 2-17, 2-18
— Fehlermeldungen 2-55
— Fixpunkte 6-59, 6-61
— Listen 2-23
Ausgabe-Systemdatei 5-1ff
Ausgabemöglichkeiten 2-17, 2-18
Autolink-Mechanismus 4-7

Bandverarbeitung 5-5, 5-7
Bearbeitung
— von Dateien 5-5
— von Systemdateien 5-3
Bedienungsplatz 5-1, 6-21
Beendigungsverhalten
— des COB1 2-58
— des COBOL-Programmes 5-8
Benutzerschalter 5-16ff
Bereitstellung des Quellprogramms 1-1
Betriebsmittelzuweisung,
— Ausgabe des Übersetzers 2-18
— Bandverarbeitung 5-7
— Eingabe in den Übersetzer 2-3
— Plattenverarbeitung 5-5
— Systemdateien 5-1ff
Bibliothek
— LMS 1-19
Bibliotheksabschnitte 2-8
Bibliothekselement 1-19, 2-9, 3-2
Bibliotheksname (LMS-) 2-4, 2-9
Bindemodul 2-1, 2-17, 2-22, 3-1, 4-1, 6-7
Bindemodulbibliothek 2-12, 2-22, 3-1, 5-8
Bindemoduldatei 2-17, 2-22, 3-1, 4-1, 5-8
Binden 2-1
Binder 4-1ff
Block (PAM-) 6-28, 6-38
Blockstruktur 6-45
Blockteilung 6-46

CALL-Anweisung 6-13
chained I-O 6-28, 6-40
CLOSE-Anweisung 6-31, 6-32
COBLIB (Dateikettungsname) 2-5, 2-11, 2-12, 2-57
COBOL-Bibliotheken 2-3, 2-8
COBOL-Bibliothekstext 2-12
COBOL-DML A-26
COBOL-Quellprogramm 1-2, 1-3
COBOL Return-Code, sh. interner Return-Code
COBOL-Verben 5-14
COBRUN-Anweisungen 2-13ff, 2-22, 2-24, 2-55, 2-62
COBRUN-Operanden 2-13ff, 2-22, 2-24, 2-25, 2-55, 2-57, 2-62, 6-35
COB1-Ablaufzeitsystem 4-1, A-19
COMPILER-INFO 5-22
CONSOLE 6-22
COPY-Anweisung 2-10ff
COPY-Element 1-19
COPY-Kommando 1-4
CPU-TIME 5-22

DATA DIVISION CODE 2-34, 2-39
DATA-Kommando 1-4, 1-5
Database Handler (DBH)
— independent DBH A-26
— linked-in-DBH A-26
Dateiaufbereiter EDT 1-4
Dateien 1-2, 1-4, 5-5, 6-23
Dateikettungsname 2-5, 2-10, 5-5

Dateiname 2-4, 5-2, 6-27, 6-47
Dateiorganisation 5-5, 6-23
Dateistruktur 6-44
Dateitypen 1-2
Datenbank A-26
Datenblock 6-45, 6-50 ff
Datenkonventionen bei Prog.verknüpfung 6-14
Datenstation 1-1, 1-2, 2-2, 2-7
Datenverwaltungssystem (DVS) 6-23, 6-49, 6-56
Deadlock 6-52, 6-57
DIAGNOSTIC LISTING 2-17, 2-19, 2-24, 2-55 ff
DIAGTEXT (COBRUN-Operand) 2-16, 2-55, 2-57, 2-62
Dialog 1-1, 1-4, 1-5, 2-3, 2-7, 5-2, 5-3
Dialogtesthilfe AID 5-11
direkte Eingabe 1-2, 2-5, 2-6
— eines Quellprogramms im Stapelbetrieb 2-5
— eines Quellprogramms im Dialogbetrieb 2-6, 2-7
DISPLAY-Anweisung 5-2, 6-21
DLL 4-2 ff
DML-Anweisung A-23, A-26
DO-Prozedur 5-16
DPAGE (Dienstprogramm) 6-39
Druckervorschub
— nach Lochbandkanälen 6-29
— um Zeilenzahl 6-29
DUMMY 5-2, 5-3, 5-5
Dynamischer Bindelader (DLL) 4-2 ff
dynamischer Zugriff 6-24

EAM-Ausgabedatei 2-22, 5-3
EAM-Bindemoduldatei* 2-22, 4-1, 4-4
EDT 1-4, 1-5
Ein-Ausgabe-Zustände 6-31, 6-40 ff, 6-48
Eingabe für DLL 4-4
Eingabe in LMS-Bibliothek 1-19
Eingabe in Datei 1-4
— Quellprogramm im Kartenformat 1-4
— Quellprogramm, mit EDT 1-5
Eingabe in den Übersetzer 2-2
— aus Bibliotheken 2-3, 2-5, 2-9 ff
— Quellprogramme 2-5 ff
— Quellprogrammteile 2-10
Eingabe über SYSDTA 2-4
Eingabe-Systemdatei 5-1
Eingabemöglichkeiten 2-2
Einsprung in Upro 6-13
Element (LMS) 1-19, 2-9, 3-2
Elementname 2-9, 3-2
END-Anweisung (COBRUN) 2-13, 2-62
END-Kommando 1-4
ENTER-Datei 2-3, 2-4, 5-3, 5-18
ENTER-Kommando 2-5, 5-18
Entsperren von Datenblöcken 6-49
ERASE-Kommando 2-22
ERDICT (COBRUN-Operand) 2-16, 2-57, 2-62
Eröffnungsart 6-26
ERRLINK 2-20, 2-59
ERRLST (COBRUN-Operand) 2-15, 2-57, 2-62
ERRPRn (COBRUN-Operand) 2-16, 2-24, 2-57, 2-62
EXECUTE-Kommando 2-6, 2-58, 4-3, 5-9 ff
EXHIBIT-Anweisung 5-1, 6-3, 6-21
Explizites Binden 4-6
External Reference (ER) 2-31

Fehlerdatei 2-20, 2-55, 2-59
Fehlerklassen 2-56
Fehlermeldungslisten 2-17, 2-19, 2-55
Fehlermeldungszeilen 2-55
FILE STATUS 6-31, 6-40 ff, 6-48, 6-51
FILE-Kommandos 2-3, 2-5, 2-20, 2-57, 5-5, 5-7, 6-27, 6-40, 6-47
Fixpunkt 6-59, 6-60
Fixpunktdatei 6-59, 6-60
Floppy Disk 5-2
Freigeben von Datenblöcken 6-49
Füllungsgrad 6-26 a, 6-47

gekettete Ein-Ausgabe 6-27, 6-40
Grundsegment 6-7

Hauptmodul 6-9

Implizites Binden 4-6
independent DBH A-26
Indexblock 6-44
Indexeintragung 6-44
indirekte Eingabe 1-2, 2-2, 2-3
— aus Bibliothek 2-9
— von Quellprogramm-Datei 2-7, 2-8
— aus LMS-Programmbibliothek 2-10
Indizierte Dateiorganisation 5-5, 6-24, 6-44
Initialisierung eines Stapelprozesses 1-5
Internadreßbuch (ISD) 2-31
interner Return-Code 5-8
ISAM 1-4, 2-3, 6-23
ISAM-Schlüssel 1-5, 6-44 ff
ISO-7-Bit Code 6-30

Jobvariablen
— Kommunikation über 5-19 ff
— Rückkehrcode-Anzeige in 2-58, 5-8

Kartenleser 5-2, 5-3, 6-21
Kartenstanzer 5-2, 5-3
katalogisierte Bindemodulbibliothek 3-1, 4-4, 5-8
katalogisierte Datei 2-17, 2-18, 2-19, 5-2, 5-5
Klasse-4-Speicher 6-10
Klasse-6-Speicher 6-10
Konventionen
— Assembler 6-16, 6-19
— Unterprogrammtechnik 6-14
Kopieren von Dateien 1-4

Label Definition (LD) 2-31
Lademodul 2-1, 4-2, 4-7, 5-8
Laden 4-3, 4-4, 5-8, 5-10
Leersätze 6-38
LINEnn (COBRUN-Operand) 2-15, 2-57, 2-63
LINK (COBRUN-Operand) 2-14, 2-22, 2-57, 2-63
Linkname 2-20, 5-5, 6-23
— COBLIB 2-5
— ERRLINK 2-20, 2-59
— LIBnnn 1-20
— LOCLINK 2-20,2-60
— OBJLINK 2-20, 2-60
— SORTCKPT 5-6, 6-59
— SORTIN, SORTINxx 5-6, 6-59
— SORTOUT 5-6, 6-59
— SORTWK 5-6, 6-58
— SRCLIB 2-5, 2-10, 2-13, 2-57, 2-64
— SRCLINK 2-20, 2-60
— SYSnnn 6-60
— SYSnnnA, SYSnnnB 6-60
— *linkname 5-19
Listenausgabe 2-13, 2-17, 2-23
Listenteil
— Datenteil (DATA DIVISION) 2-34, A-22
— Prozedurteil (PROCEDURE DIVISION) 2-36, A-22
Listenzeilen
— LINKAGE SECTION 2-50
— WORKING-STORAGE SECTION 2-50
LMR (Dienstprogramm) 3-1
LMS (Softwareprodukt) 1-19, 2-10, 3-2
LOAD-Kommando 4-4, 5-8
LOCATOR/MAP LISTING 2-17, 2-19, 2-24, 2-51
Lochkarte 1-4
LOCLINK 2-20
logische Fehler 5-11
Logischer Kontrollabschnitt 0 (LC0) A-22
Logischer Kontrollabschnitt 1 (LC1) A-22
Logischer Kontrollabschnitt 2 (LC2) A-22
Logischer Kontrollabschnitt 3 (LC3) A-22
Logischer Kontrollabschnitt 4 (LC4) A-22
Logischer Kontrollabschnitt 5 (LC5) A-22
Logischer Kontrollabschnitt 6 (LC6) A-22
Logischer Kontrollabschnitt 7 (LC7) A-22
Logischer Kontrollabschnitt 8 (LC8) A-22
LOW#UP (COBRUN-Operand) 2-14, 2-22, 2-57, 2-63
LSD-Namen 5-13

MAPALL (COBRUN-Operand) 2-15, 2-57, 2-63
MAPSRT (COBRUN-Operand) 2-15, 2-24, 2-50, 2-63
Mehrfachbenutzbare Programme 6-7
Mehrfachbenutzbarkeit des Ablaufzeit-Systems 6-12
Meldungen des Übersetzers 2-15, 2-17, A-1 ff
MERGE-Anweisung 6-58
Metasprache 1-1
— für BS2000-Anwendung 1-1
— für COBOL-Formate 1-1
Mischen 6-58
MODULE (COBRUN-Operand) 2-15, 2-18, 2-19, 2-22, 2-57, 2-63, 4-2
Modulbibliothek 3-2, 4-5
MULTIPLE FILE TAPE-Klausel 5-7

NESTPF (COBRUN-Operand) 2-14, 2-57, 2-63
NODDLIST (COBRUN-Operand) 2-16, 2-24, 2-26, 2-57, 2-63
NOCOPY (COBRUN-Operand) 2-16, 2-57, 2-63

OBJECT PROGRAM LISTING 2-17, 2-19, 2-31, 2-32 ff
OBJECT-COMPUTER-Paragraph 6-1
Objektmodul 2-17, 2-19, 4-1, 4-4, 4-7, A-23
Objektmodulbibliothek 3-1
Objektmodulformat A-22
Objektprogramm 2-19, 2-22, 4-1
Objektprogrammliste 2-17, 2-19, 2-24, 2-31, 2-60
OBJLINK 2-20
OPEN-Anweisung 6-23, 6-26, 6-37
Operandenadreßliste 6-15
Organisationsformen von Dateien 5-5, 6-23 ff, 6-26

PAM-Block 6-28, 6-38, 6-40
PAM-Schlüssel 6-38, 6-45
PAM-Seite 6-38, 6-45
PARAMETER-Kommando 2-13, 2-19 ff, 2-57, 2-59
PARAM8 (COBRUN-Operand) 2-14, 2-57, 2-63
Plattenspeicher(datei) 6-23, 6-27, 6-32, 6-37, 6-44

Primärzuweisung
— bei Systemdateien (Standardzuweisung) 2-6, 5-2
— im SPACE-Operanden (FILE-Kommando) 6-28, 6-40, 6-47
PRINT-Kommando 1-5, 2-17, 2-21, 6-30
PRINTERDOD 6-28
PROCEDURE DIVISION CODE 2-35, 2-51
PROCESS-INFO 5-22
Programmaufruf 5-8ff
Programmierbare Testhilfen 6-1
Programmverknüpfung
— Assembler-COBOL 6-16, 6-19
— COBOL-Assembler 6-16
— COBOL-COBOL 6-13
Prozeß 1-4, 5-2, 5-16
Prozeßschalter 5-16
Pseudodatei (*DUMMY) 5-2ff
Puffer 6-28, 6-40, 6-47
Pufferlänge 6-28, 6-40, 6-47

Quelldaten 2-2, 2-10
Quelleingabe 2-2
Quellprogramm in Datei eingeben 1-1, 1-4
Quellprogramm-Datei 1-2
Quellprogrammliste 2-19, 2-21, 2-23ff
Quellprogrammlistenzeile 2-27
Quellprogrammteile 2-2, 2-10
Querverweisliste 2-19, 2-24, 2-50ff
QUOTE1 (COBRUN-Operand) 2-13, 2-57, 2-63

RANGECHECK (COBRUN-Operand) 2-13, 2-64
RCARD-Kommando 1-5
READ-Anweisung 6-26ff, 6-37, 6-44, 6-50
READY TRACE-Anweisung 6-5
RECORD KEY 6-44
reentrant Code 6-7
Registerbenutzung 6-15ff
Registerkonventionen 6-15
relative Adressierung 6-37
Relative Dateiorganisation 5-5, 6-23ff, 6-37
RELATIVE KEY 6-37
relative Satznummer 6-39
relativer Satzschlüssel 6-39
RELEASE-Kommando 5-6
RERUN-Klausel 6-59, 6-61
RESET TRACE-Anweisung 6-5
RESTART-Kommando 6-62
RESUME-Kommando 5-8
RETURN-CODE 6-15
Rückkehrcode-Anzeige in Jobvariablen 2-58, 5-8
Rücksprung aus Upro 6-14
Runtimesystem (RTS) 6-12

SAM 1-4, 2-3, 6-23
Satzbezeichner 6-33
Satzformate 6-24
Schalter
— Benutzer- 5-16ff
— Prozeß- 5-16ff
Schema A-26
Schlüsselwörter (PARAM-Kommando) 2-19, 2-57, 2-59
Schnelldrucker 2-18
Segmente des COB1-Übersetzers A-18
Segmentierung 6-6, 6-7ff
segmentiertes Programm 6-7ff
Seitenwechsel 2-26
Sekundäreingabe 2-2
Sekundärzuweisung (SPACE-Operand im FILE-Kommando) 6-28, 6-36, 6-40, 6-47, 6-58
SELECT-Klausel 5-6, 6-39
SEMCHK (COBRUN-Operand) 2-13, 2-22, 2-57, 2-64
SEQERR (COBRUN-Operand) 2-16, 2-57, 2-64
Sequentielle Dateiorganisation 5-5, 6-23, 6-27ff
sequentieller Zugriff 6-24
SEVERITY CODE 2-24, 2-56
SHARE-Kommando 6-7, 6-12
shareable 6-7, 6-12
Shared Code 6-10
SHARUPD (FILE-Kommando) 6-40, 6-47, 6-49ff, 6-55
Sicherstellung von Bindemoduln 3-1
Sicherstellungsbereich 6-15
Simultanverarbeitung 6-49ff
Sonderregister
— SORT 6-58
— TALLY 6-2
— RETURN-CODE 6-15
SORT (Softwareprodukt) 6-58
SORT-Anweisung 6-58
SORT-Sonderregister 6-57
Sortierdatei 6-58
Sortieren 6-58
SOURCE LISTING 2-17, 2-19, 2-27ff
SOURCE-COMPUTER-Paragraph 6-1
SPECIAL-NAMES-Paragraph 5-2, 6-21
Sperren von Datenblöcken 6-49ff
SPOOLIN-Datei 1-2, 1-4, 2-3, 5-3
SPOOLOUT-Datei 5-3
SRCELEM (COBRUN-Operand) 2-5, 2-10, 2-13, 2-57, 2-64
SRCLIB 2-5, 2-10, 2-13, 2-57, 2-64
SRCLINK 2-20
SSEQ # GEN (COBRUN-Operand) 2-14, 2-57, 2-64
Standardblock 6-28, 6-40, 6-47
Standardzuweisung (FILE) 6-47
Stapel 1-4, 2-3, 2-4, 5-3

Starten 4-3, 4-4, 5-8ff
Statischer Binder 4-2, 4-7
STEP-Kommando 2-19
Steueranweisungen
— COB1 2-2, 2-13
Steuerinformation für Druckerdateien 6-28
Steueranweisungsliste 2-25
STOP literal-Anweisung 5-1, 6-21
Subschema A-26
SYMTEST (COBRUN-Operand) 2-14, 2-57, 2-64
SYNCHK (COBRUN-Operand) 2-13, 2-57, 2-65
SYSFILE-Kommando 2-4, 2-18, 2-57, 4-4, 5-2, 5-4
Systembindemodulbibliothek 4-4
Systemdatei 5-1, 6-21
— SYSCMD 2-4
— SYSDTA 2-4, 2-15, 3-1, 5-1ff, 6-21
— SYSIPT 5-1ff, 6-21
— SYSLST 2-21, 5-1ff, 6-21
— SYSLSTnn 5-1ff
— SYSOPT 5-1ff, 6-21
— SYSOUT 3-1, 5-1ff, 6-21

TABS-Anweisung (EDT) 1-6
TALLY (Sonderregister) 6-2
TCBENTRY (COBRUN-Operand) 2-15, 2-57, 2-65
temporäre Bindemoduldatei * 2-17, 2-19, 2-22, 4-1, 5-8
temporäre Datei 2-17, 2-18
TERMINAL 6-21
TERMINAL-INFO 5-22
Testhilfen 5-11, 6-1
Testüberwachung 6-5
Testhilfezeile 6-1
TRACE-Anweisung 5-1, 6-5, 6-21
TRUNCATE (COBRUN-Operand) 2-15, 2-57, 2-65
TSOSLNK (Dienstprogramm) 4-2, 4-6, 4-8
TYPE-Kommando 2-6

Überlagerungssegment 6-7
Überlauf-Blöcke 6-26a
Übersetzer-Eingabe 2-2
Übersetzung, Quellprogramm 2-1
Umsetzroutinen 1-4
Umweisung (Systemdateien) 5-3
Universelles Datenbanksystem (UDS) A-26
Unterprogramm-Einsprungpunkte A-25
Unterprogrammaufruf 6-13, A-24
Unterprogramme für komplexe COBOL-Anweisungen A-19
UPAM 6-23
USING 6-13

Vorschubinformation 6-28

Wahlfreier Zugriff 6-24
Wiederanlauf 6-59, 6-61
Wiederholungsschleife 6-51
WITH DEBUGGING MODE-Klausel 6-1
WRITE-Anweisung
— COBOL 6-26ff, 6-37
— EDT 1-6
WRLST (COBRUN-Operand) 2-16, 2-17, 2-21, 2-57, 2-65
Wurzelsegment 6-7

Zugriff (COB1) 6-25
Zugriffsart 6-24
Zugriffskoordinierung 6-49
Zugriffsmethode des DVS 5-5, 6-23
Zuweisung von Betriebsmittel,
— Übersetzer-Ausgabe 2-17ff
— Systemdatei 5-1
— Datei 5-1, 5-5



Von

Name ....

Firma ....

Straße ....

Postleitzahl/Ort ....

Telefon ....

....

An
Siemens AG

K D ST QM 2
Manualredaktion

Otto-Hahn-Ring 6
8000 München 83

Leserzuschrift zum Manual
COB1 (BS2000)
COBOL-Compiler Benutzerhandbuch
Nachtrag August 1986 (Softwareprodukt COB1 V2.3A)

Seite	Kritik / Anregungen / Korrekturen

COB1 (BS2000) V2.3A Benutzerhandbuch U254-J2-Z55-3

Seite	Kritik / Anregungen / Korrekturen

Von

Name ....

Firma ....

Straße ....

Postleitzahl/Ort ....

Telefon ....

....

An
Siemens AG

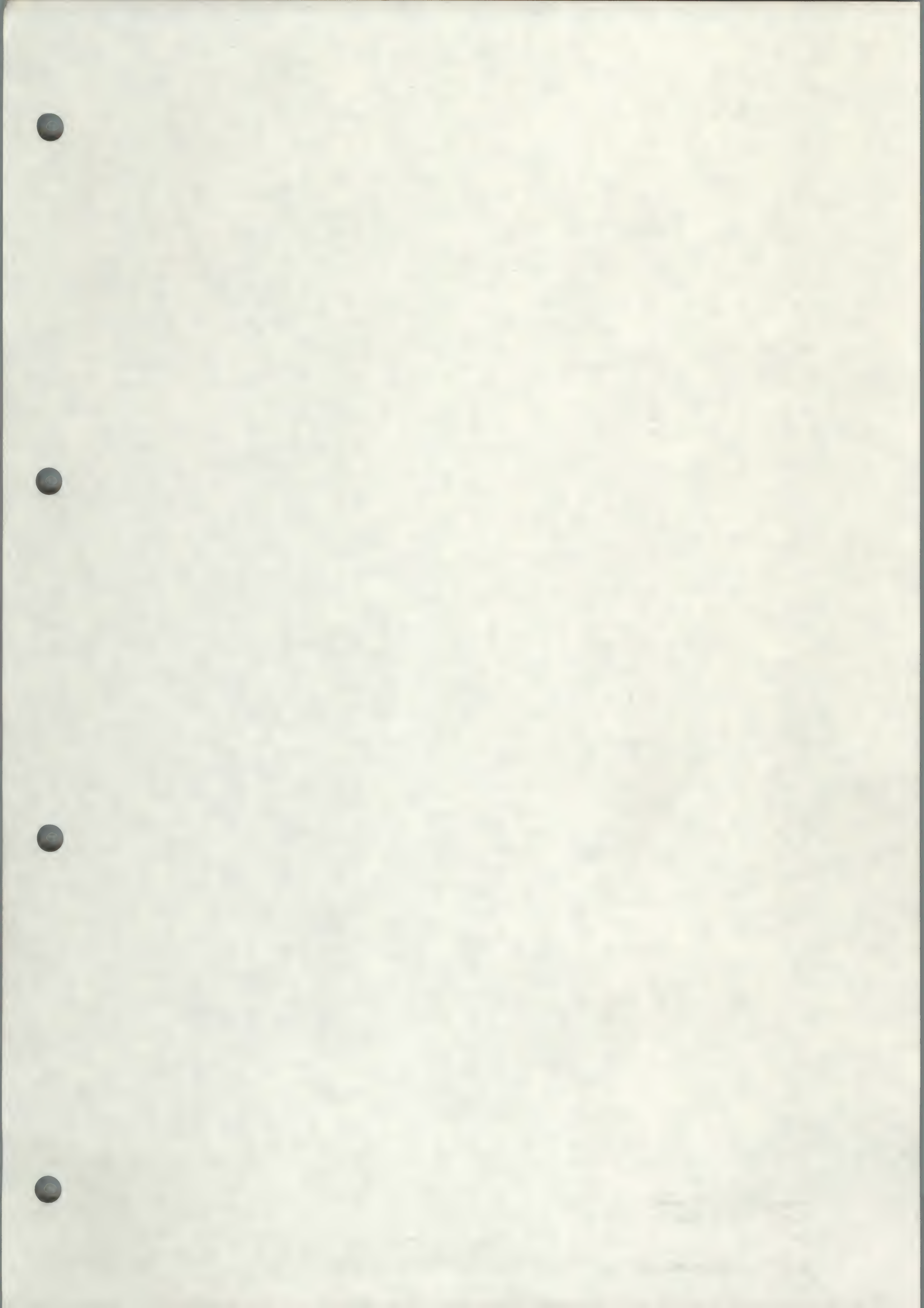
K D ST QM 2
Manualredaktion

Otto-Hahn-Ring 6
8000 München 83

Leserzuschrift zum Manual
COB1 (BS2000)
COBOL-Compiler Benutzerhandbuch
Nachtrag August 1986 (Softwareprodukt COB1 V2.3A)

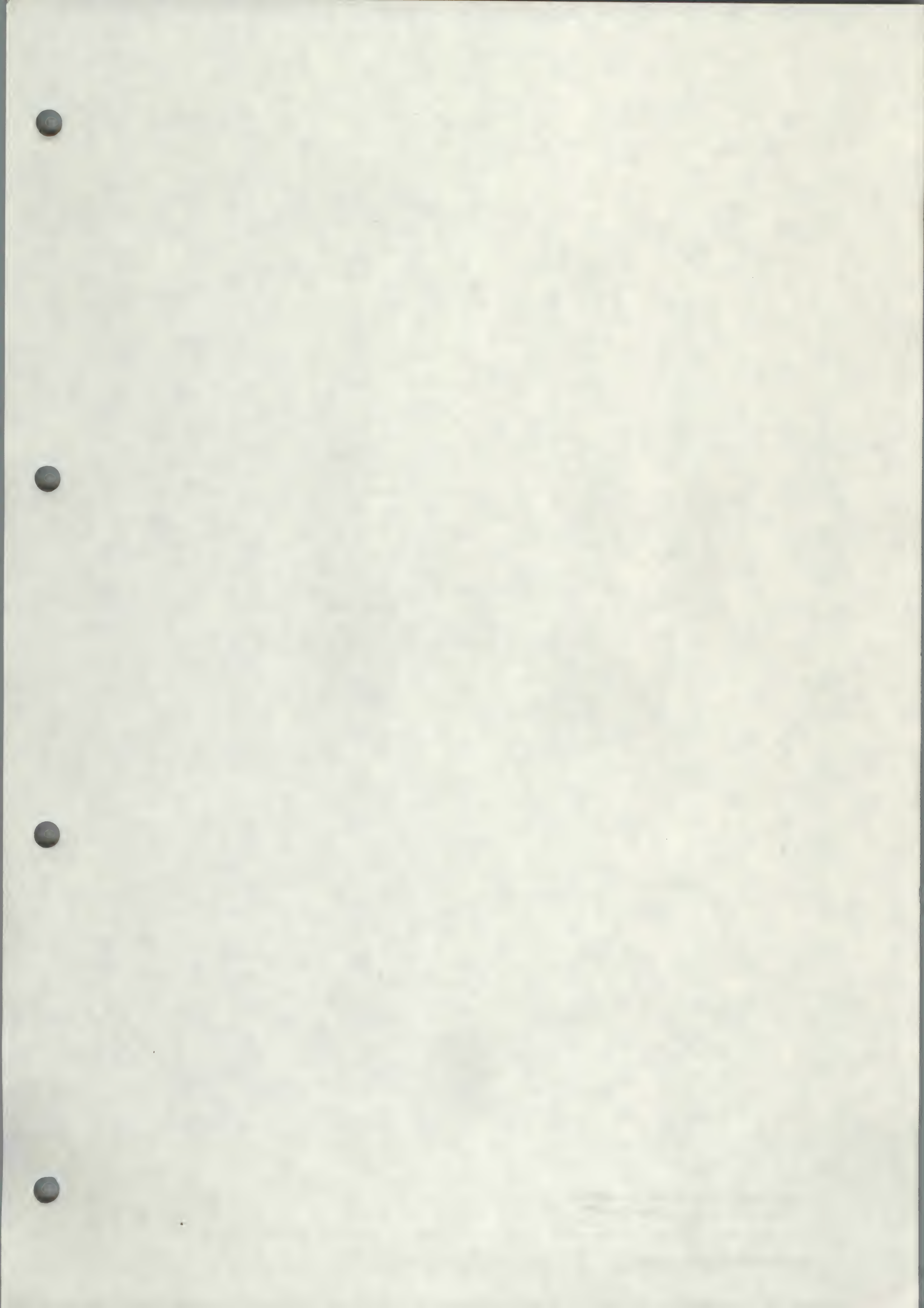
Seite	Kritik / Anregungen / Korrekturen

Seite	Kritik / Anregungen / Korrekturen



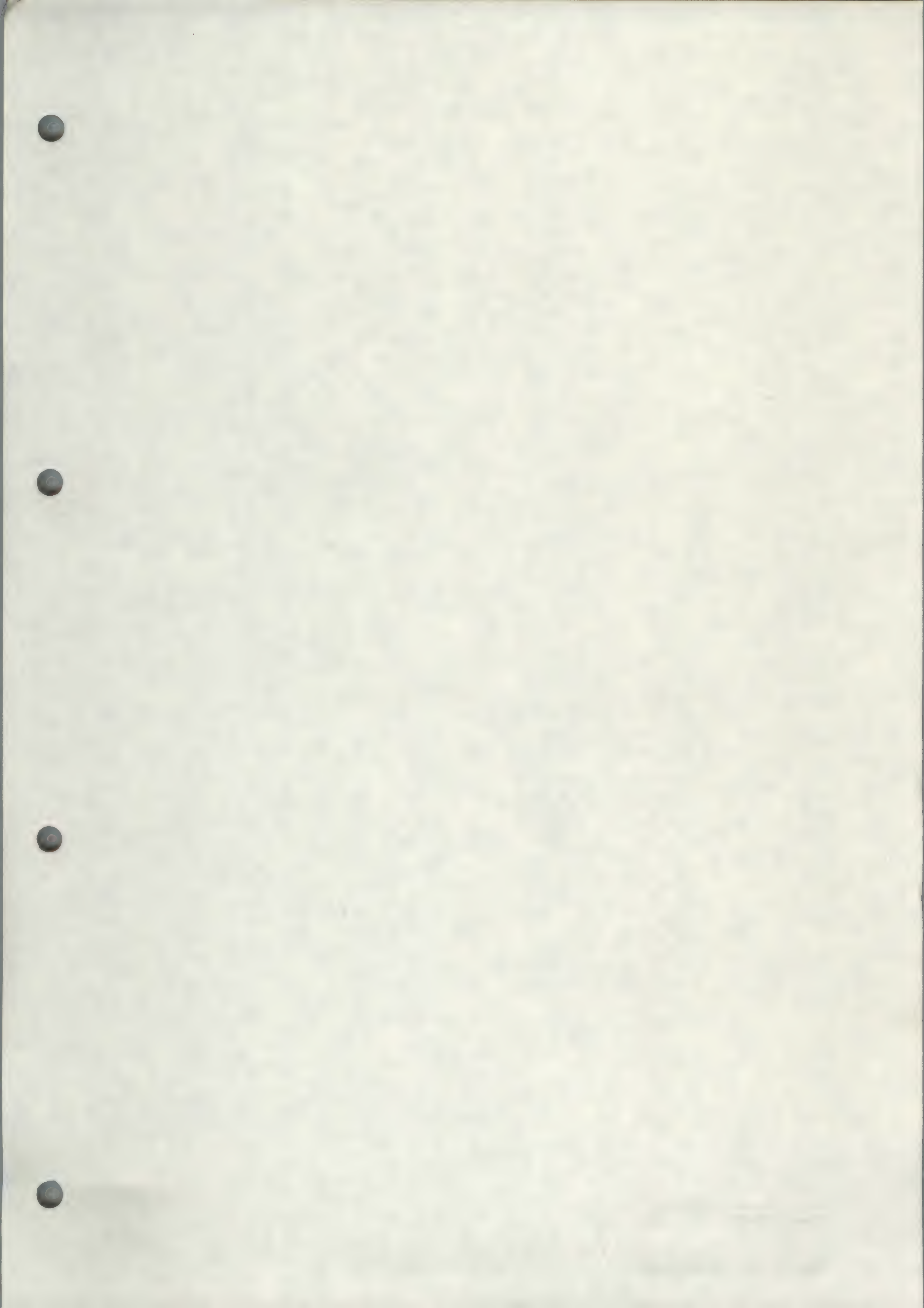
Herausgegeben vom Bereich Datentechnik
Postfach 83 09 51, D-8000 München 83

Siemens Aktiengesellschaft



Herausgegeben vom Bereich Datentechnik
Postfach 83 09 51, D-8000 München 83

Siemens Aktiengesellschaft



Herausgegeben vom Bereich Datentechnik
Postfach 83 09 51, D-8000 München 83

Siemens Aktiengesellschaft

```
        WORKING-STORAGE SECTION.\
        77  ZAHL PIC 9.\
        77  ERGEBNIS PIC ZZ9.\
        77  I PIC 99.\
        PROCEDURE DIVISION.\

        ANFANG.\                                                        ②
            DISPLAY "ZWEISTELLIGE ZAHL EINGEBEN (ENDE BEI 0):" UPON TERMI\
        — NAL.
            ACCEPT ZAHL FROM TERMINAL.\
            IF ZAHL NOT NUMERIC\
            DISPLAY "EINGABE ZAHL MUSS NUMERISCH SEIN" UPON TERMINAL\
            GO TO ANFANG.\
            IF ZAHL = ZERO STOP RUN.\
            PERFORM RECHNEN VARYING I FROM 1 BY 1 UNTIL I > 10.\
            GO TO ANFANG.\
        RECHNEN.\
            MULTIPLY I BY ZAHL GIVING ERGEBNIS.\
.......................................................................
H!H                                                                     ③
"DATEI QUELL.FLEXI GESCHLOSSEN"\
EDOR BEENDET
```

① Die Datei QUELL.FLEXI wird eröffnet. Anschließend zeigt EDOR den Dateianfang am Bildschirm.
Anmerkung: Falls QUELL.FLEXI eine SAM-Datei wäre, müßte man statt dessen eine Hilfsdatei eröffnen (z. B. O,HILF,N) und den Inhalt von QUELL.FLEXI hineinkopieren (z. B. K200,QUELL.FLEXI).

② Die Änderungen werden am Bildschirm mit Hilfe der Tastatur durchgeführt. Der zu löschende Satz wird durch einen Satzbegrenzer, den Gegenschrägstrich (\ ), ersetzt. Anschließend müssen die veränderten Datensätze eingegeben werden.

③ Mit der Anweisung „– –" könnte man kontrollieren, ob die Änderungen wie gewünscht in die Datei übernommen wurden.
Die Anweisung „H" schließt die Datei, das zweite „H" beendet EDOR.

1.3 COBOL-Bibliotheken (COBLUR)

1.3.1 Übersicht über die Bereitstellung in COBOL-Bibliotheken

Eine COBOL-Bibliothek ist eine ISAM-Datei, in der COBOL-Quellprogramme und -Quellprogrammteile in komprimierter Form abgespeichert werden können. Elemente einer solchen Bibliothek werden vom Übersetzer eingelesen, wenn er eine BASIS-Anweisung erkennt oder eine COPY-Anweisung [1] (siehe Abschnitt 2.2) in dem zu übersetzenden Programm vorfindet. Der Inhalt einer COBOL-Bibliothek bleibt so lange verfügbar, bis der Benutzer ihn ändert oder löscht. Dazu benötigt er — ebenso wie zu jedem anderen Zugriff zu der Bibliothek — das Dienstprogramm COBLUR [3], das Aufbau und Wartung einer COBOL-Bibliothek ermöglicht.

Im Vergleich zum Einsatz von Quellprogramm-Dateien erweist sich das Speichern in einer COBOL-Bibliothek als platzsparend. Andernfalls benötigt jedes noch so kleine COBOL-Quellprogramm eine eigene Datei, von der jede mindestens 3 PAM-Seiten auf dem Datenträger belegt. Der Einsatz von COBOL-Bibliotheken bietet sich vor allem bei der Entwicklung von Programmsystemen an, bei denen die einzelnen Programme teilweise gleiche Unterroutinen oder Dateibeschreibungen benutzen.

1.3.2 Struktur von COBOL-Bibliotheken

Eine COBOL-Bibliothek ist eine ISAM-Datei mit Sätzen variabler Länge, die einen 14 Zeichen langen ISAM-Schlüssel am Satzanfang besitzen.

Die Bibliothek unterteilt sich in 4 **Bibliotheksabschnitte:**

Teil 1 enthält Einträge für den Erkennungs- und Maschinenteil (IDENTIFICATION und ENVIRONMENT DIVISION) von Quellprogrammen,

Teil 2 enthält Einträge für den Datenteil (DATA DIVISION) von Quellprogrammen,

Teil 3 enthält Einträge für den Prozedurteil (PROCEDURE DIVISION) von Quellprogrammen,

Teil 4 enthält vollständige Quellprogramme oder Programmteile.
(Bei COPY- oder BASIS-Anweisungen greift der Übersetzer immer auch auf diesen Teil zu.)

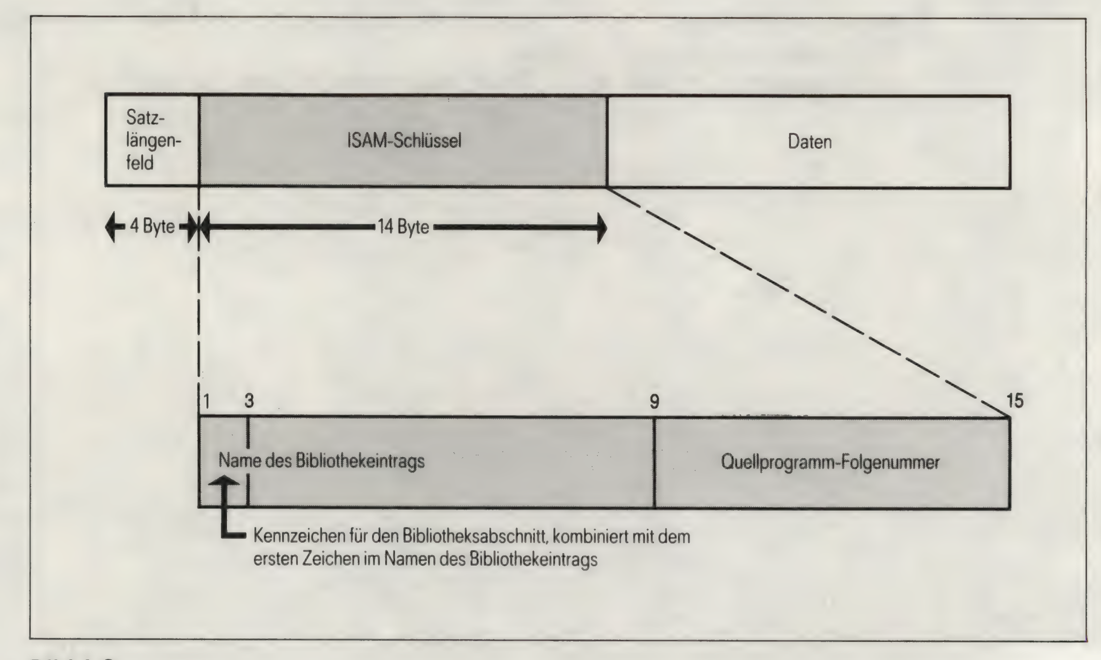


Bild 1-2

Aufbau des ISAM-Schlüssels für die Datensätze einer COBOL-Bibliothek

1.3.3 Eingabe in COBOL-Bibliotheken

Mit Hilfe des Dienstprogramms COBLUR [3] gibt der Benutzer Quellprogramme bzw. Quellprogrammteile in eine COBOL-Bibliothek ein. Die Eingabe kann erfolgen:

- über die Systemdatei SYSDTA, d.h. wahlweise von einer Datenstation, Datei oder dem Lochkartenleser, im Stapelbetrieb auch aus der SPOOLIN-Datei.
- aus einer anderen COBOL-Bibliothek.

Das Programm COBLUR wird dazu mit dem EXECUTE-Kommando aufgerufen und erwartet dann COBLUR-Steueranweisungen von der Systemdatei SYSDTA.

Für die **Eingabe von Quellprogrammen oder Quellprogrammteilen** sind folgende **COBLUR-Anweisungen** von Bedeutung (die dafür unbedingt erforderlichen sind durch Unterstreichung hervorgehoben):

<u>COBLIB</u> vereinbart den Namen der COBOL-Bibliothek, die COBLUR neu aufbauen bzw. bearbeiten soll.

SOURCE gibt an, woher die einzufügenden Einträge kommen sollen. Standardmäßig ist SYSDTA vorgesehen, d.h. in diesem Fall kann die Anweisung SOURCE weggelassen werden.

COPYLIB wird beim Kopieren aus einer anderen COBOL-Bibliothek benötigt.

<u>CATALS</u> legt fest, wo die Einträge in der COBOL-Bibliothek zu machen sind und gibt den Namen des neuen Bibliothekseintrags an.

DSPLYE ermöglicht, das Inhaltsverzeichnis der COBOL-Bibliothek oder den Inhalt von Bibliotheksabschnitten zu protokollieren.

<u>END</u> beendet den COBLUR-Lauf

Alle COBLUR-Anweisungen beginnen mit einem Leerzeichen (X'40').

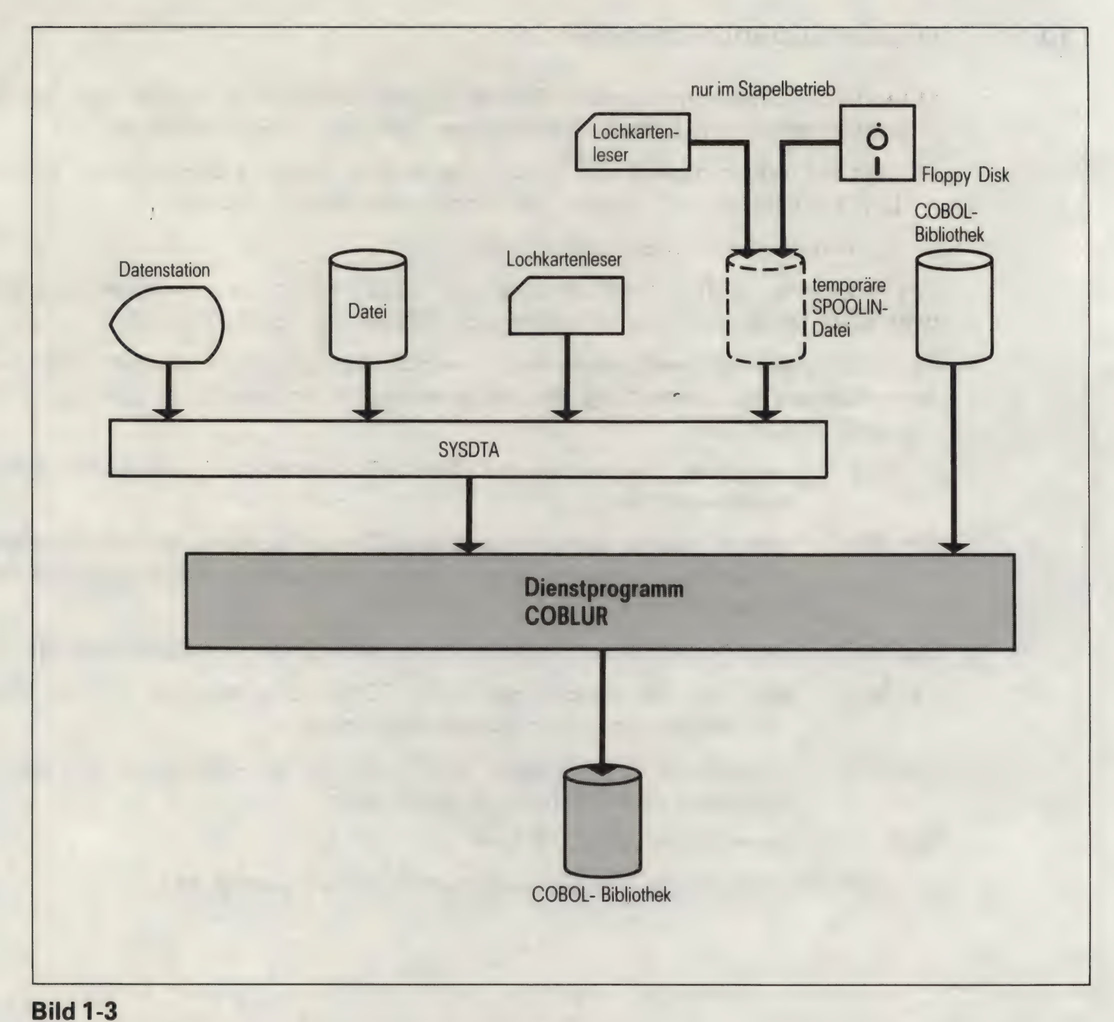


Bild 1-3

Eingabemöglichkeiten für eine COBOL-Bibliothek

Im folgenden werden zwei typische Anwendungsfälle im Stapel- und im Dialogbetrieb anhand je eines Beispiels vorgestellt.

Beispiel 7: Eingabe in eine COBOL-Bibliothek im Stapelbetrieb

```
/LOGON...
/EXEC $COBLUR                                                    ①
   COBLIB BIB. 1                                                 ②
   CATALS C4,LK                                                  ③
Quellprogramm
   DSPLYE CD                                                     ④
   DSPLYE C4,LK
   END                                                           ⑤
/LOGOFF
```

① Das Dienstprogramm COBLUR wird aufgerufen.

② Als Name der COBOL-Bibliothek, die COBLUR bearbeiten soll, wird BIB. 1 vereinbart.

③ In Teil 4 der COBOL-Bibliothek wird das nachfolgende Quellprogramm unter dem Namen LK eingetragen. Jede Quellprogrammzeile (je 80 Byte) wird zu einem Satz in der Bibliotheksdatei.

④ Das Inhaltsverzeichnis der Bibliothek (CD) und der Eintrag LK aus Teil 4 werden in die Systemdatei SYSLST (Schnelldrucker) ausgegeben.

⑤ Der Lauf von COBLUR wird beendet.

Beispiel 8: Eingabe in eine COBOL-Bibliothek im Dialogbetrieb

```
/ EXEC $COBLUR                                                    ①
% P500 LOADING
PROGRAM COBLUR/10.0 STARTED
* COBLIB BIB.1                                                    ②
OLD COBLIB= FILE OPENED.
* CATALS C3,ABCOB,SEQNCE,SAVE                                     ③
123456 A   B
* 000010    ADD 500 TO H-FELD1.                                   ④
* 000020    MOVE H-FELD1 TO H-FELD2.
* 000030    MOVE SPACES TO PR-SATZ.
* 000040    WRITE PR-SATZ FROMH-FELD2 AFTER 3.
* DSPLYE C3,ABCOB,SO                                              ⑤
PARTITION 3
ABCOB   000010    ADD 500 TO H-FELD1.
ABCOB   000020    MOVE H-FELD1 TO H-FELD2.
ABCOB   000030    MOVE SPACES TO PR-SATZ.
ABCOB   000040    WRITE PR-SATZ FROMH-FELD2 AFTER 3.
* CATALS C4,BEISP1                                                ⑥
123456 A   B
*                                                                 ⑦
/ SYSFILE SYSDTA=QUELL.EINXEINS                                   ⑧
/ R
        IDENTIFICATION DIVISION.
        PROGRAM-ID. EINXEINS.
               .
               .
               .
ISSUE /SYSFILE SYSDTA=(PRIMARY) AND /RESUME
BKPT PCOUNT 0011E4
/ SYSFILE SYSDTA=(SYSCMD)                                         ⑨
/ R
* DSPLYE C4,BEISP1                                                ⑩

PROGRAM COBLUR/10.0 STARTED

COBLIB=BIB.1
PARTITION 4
BEISP1  000010 IDENTIFICATION DIVISION.
BEISP1  000020 PROGRAM-ID. EINXEINS.
                 .
                 .
                 .
* END                                                             ⑪
COBLIB= FILE CLOSED.
```

① Das Dienstprogramm COBLUR wird aufgerufen.

② Der Name BIB.1 der zu bearbeitenden COBOL-Bibliothek wird angegeben.

③ Unter dem Namen ABCOB soll in den Teil 3 der COBOL-Bibliothek ein neuer Eintrag erstellt werden. SEQNCE bewirkt ein Prüfen des Schlüssels, der wegen SAVE übernommen wird.

④ Vier Quellprogramm-Sätze werden in die COBOL-Bibliothek eingegeben.

⑤ Der Eintrag ABCOB in Teil 3 wird auf SYSOUT (Angabe SO), d.h. auf der Datenstation protokolliert.

⑥ In Teil 4 der aktuellen COBOL-Bibliothek soll unter dem Namen BEISP1 ein Quellprogramm übernommen werden. Dieses Programm ist bereits in der Datei QUELL.EINXEINS.FEHLER vorhanden.

⑦ Mit der EXCAPE-Funktion der Datenstation (Datenübertragungstaste) wird COBLUR unterbrochen und in den Systemmodus übergegangen. So kann im folgenden SYSFILE-Kommando SYSDTA auf die neue Eingabedatei für COBLUR umgelegt werden.

⑧ Das SYSFILE-Kommando weist die Systemdatei SYSDTA der Datei QUELL.EINXEINS.FEHLER zu, das RESUME-Kommando (Abkürzung: R) bewirkt eine Rückkehr in den Programm-Modus. COBLUR läuft weiter, d.h. es liest das Quellprogramm. Sobald das Ende der Datei QUELL.EINXEINS.FEHLER erkannt wird, sendet COBLUR die folgende Meldung und unterbricht seinen Lauf.

⑨ Die Systemdatei SYSDTA wird auf die Datenstation gelegt: (SYSCMD) oder aber (PRIMARY).

⑩ Eintrag BEISP1 aus Teil 4 der COBOL-Bibliothek soll auf SYSLST (Schnelldrucker) protokolliert werden.

⑪ Das Programm COBLUR wird beendet.

1.3.4 Änderungen in COBOL-Bibliotheken

Will man Daten in einer COBOL-Bibliothek abändern, löschen oder einfügen, so bietet COBLUR dafür Steueranweisungen an. Die damit durchgeführten Änderungen in der COBOL-Bibliothek sind nicht temporär, d.h. man muß sie von denen unterscheiden, die nur für einen einzigen Übersetzungslauf gelten (siehe Abschnitt 2.2.5).

Zusätzlich zu den **COBLUR-Anweisungen** für die Eingabe, die bereits im Abschnitt 1.3.3 aufgeführt sind, können für **Änderungen** folgende Angaben erforderlich sein:

DELETS   löscht ein Element aus einem Bibliotheksabschnitt.

Mehrere Anweisungen bilden eine in sich abgeschlossene Korrektureinheit, die mit STARTC beginnt und mit ENDC endet. Dazwischen stehen Daten, DELETE- oder INSERT-Anweisungen:

STARTC   vereinbart den zu ändernden Eintrag.

DELETE   löscht Quellprogramm-Anweisungen innerhalb des mit STARTC vereinbarten Elements und ist nur nach STARTC erlaubt.

INSERT   fügt Quellprogramm-Anweisungen in das mit STARTC vereinbarte Element ein und ist nur nach STARTC erlaubt.

ENDC   schließt die Korrektureinheit ab.

Beispiel 9: Änderungen in einer COBOL-Bibliothek

```
/ EXEC $COBLUR
%  P500 LOADING
PROGRAM  COBLUR/10.0  STARTED                                              ①
* COBLIB BIH.1
OLD COBLIB= FILE OPENED.
* DSPLYE C4,LK,SO                                                          ②
PARTITION 4
LK      000010 IDENTIFICATION DIVISION.
LK      000020 PROGRAM-ID. EINXEINS.
LK      000030 ENVIRONMENT DIVISION.
LK      000040 DATA DIVISION.
LK      000050 WORKING-STORAGE SECTION.
LK      000060 77 ZAHL       PUC 99.
                      .
                      .
                      .
LK      000220     DISPLAY I "*" ZAHL "=" ERGEBNIS UPON TERMINAL.
* STARTC C4,LK                                                             ③
*123456 A   B
* DELETE 000060                                                            ④
* INSERT 000060                                                            ⑤
*000061 77 ZAHL     PIC 99.
* ENDC                                                                     ⑥
* DSPLYE C4,LK,SO                                                          ⑦
PARTITION 4
LK      000010 IDENTIFICATION DIVISION.
LK      000020 PROGRAM-ID. EINXEINS.
LK      000030 ENVIRONMENT DIVISION.
LK      000040 DATA DIVISION.
LK      000050 WORKING-STORAGE SECTION.
LK      000061 77 ZAHL     PIC 99.
                      .
                      .
                      .
LK      000220     DISPLAY I "*" ZAHL "=" ERGEBNIS UPON TERMINAL.
  END
COBLIB= FILE CLOSED.
```

① Die zu bearbeitende COBOL-Bibliothek heißt BIB. 1.

② In Teil 4 der aktuellen Bibliothek soll der Eintrag LK auf SYSOUT (Datenstation) gezeigt werden.

③ Korrekturen sollen im Eintrag LK von Teil 4 vorgenommen werden. Damit wird der Korrekturmodus eingeschaltet, der mit der Anweisung ENDC beendet wird.

④ Satz 60 wird gelöscht.

⑤ Ab Satz 60 soll eingefügt werden, und zwar der folgende Quellprogrammsatz.

⑥ ENDC beendet den Korrekturmodus.

⑦ Zur Kontrolle wird der Eintrag LK erneut auf die Datenstation ausgegeben.

Mit COBLUR können nur Sätze eingegeben, keine einzelnen Zeichen oder Satzteile verändert werden.